Configuration Handbook

101 Innovation Drive
San Jose, CA 95134
www.altera.com

Document last updated for Altera Complete Design Suite version: 11.0
Document publication date: June 2011

© 2011 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

QUALITY
ISO 9001:2008
NSAI Certified

Contents

Volume 1: Configuration Handbook

Configuration, Design Security, and Remote System Upgrades in Stratix V Devices 6
Configuration, Design Security, and Remote System Upgrades in Stratix IV Devices 64
Configuring Stratix III Devices 136
Configuring Stratix II and Stratix II GX Devices 188
Configuring Stratix and Stratix GX Devices 296
Configuration and Remote System Upgrades in Cyclone IV Devices 356
Configuration, Design Security, and Remote System Upgrades in the Cyclone III Device Family 436
Configuring Cyclone II Devices 524
Configuring Cyclone FPGAs 596
Configuration, Design Security, and Remote System Upgrades in Arria II Devices 648
Configuring Arria GX Devices 718
Configuring Altera FPGAs 824
Configuring APEX II Devices 834
Configuring APEX 20KE & APEX 20KC Devices 902
Configuring Mercury, APEX 20K (2.5 V), ACEX 1K & FLEX 10K Devices 972

Volume 2: Configuration Handbook

Contents 1044
Additional Information 1048
Section I. FPGA Configuration Devices 1050
Enhanced Configuration Devices (EPC4, EPC8, and EPC16) Data Sheet 1052
Altera Enhanced Configuration Devices 1088
Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet 1104
Configuration Devices for SRAM-Based LUT Devices Data Sheet 1142
Quad-Serial Configuration (EPCQ) Devices Datasheet 1168
Section II. Software Settings 1200
Device Configuration Options 1202
Configuration File Formats 1210
Section III. Advanced Configuration Schemes 1216
Combining Different Configuration Schemes 1218

Volume 1: Configuration

Handbook

101 Innovation Drive
San Jose, CA 95134
www.altera.com

CF5V1-4.0

Document last updated for Altera Complete Design Suite version: 11.0
Document publication date: June 2011

© 2011 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

QUALITY
ISO 9001:2008
NSAI Certified

9. Configuration, Design Security, and Remote System Upgrades in Stratix V Devices

SV51010-1.3

This chapter contains information about the Stratix® V supported configuration schemes, instructions about how to execute the required configuration schemes, and all the necessary option pin settings. This chapter also reviews the different ways you can configure your device and explains the design security and remote system upgrade features for the Stratix V devices.

This chapter includes the following sections:

- "Configuration Features" on page 9–2
- "Power-On Reset Circuit and Configuration Pins Power Supply" on page 9–2
- "Configuration Sequence" on page 9–4
- "Configuration Schemes" on page 9–7
- "Fast Passive Parallel Configuration" on page 9–9
- "Active Serial Configuration (Serial Configuration Devices)" on page 9–16
- "Passive Serial Configuration" on page 9–28
- "JTAG Configuration" on page 9–34
- "Device Configuration Pins" on page 9–38
- "Configuration Data Decompression" on page 9–42
- "Remote System Upgrades" on page 9–44
- "Design Security" on page 9–53

Stratix V devices use SRAM cells to store configuration data. Because SRAM memory is volatile, you must download the configuration data to the Stratix V device each time the device powers up. You can configure Stratix V devices using one of four configuration schemes:

- Fast passive parallel (FPP) (×8, ×16, and ×32)
- Active serial (AS) (×1 and ×4)
- Passive serial (PS)
- JTAG

All configuration schemes use either an external controller (for example, a MAX® II device or microprocessor), a configuration device, or a download cable. For more information about the configuration features, refer to "Configuration Features" on page 9–2.

© 2011 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

Configuration Features

Stratix V devices offer decompression, design security, and remote system upgrade features. Stratix V devices can receive a compressed configuration bitstream and decompress this data in real-time, reducing storage requirements and configuration time. Design security using configuration bitstream encryption is available in Stratix V devices, which protects your designs. You can make real-time system upgrades of your Stratix V designs from remote locations with the remote system upgrade feature.

Table 9–1 lists which configuration features you can use in each configuration scheme.

Table 9–1. Configuration Features for Stratix V Devices

Configuration Scheme	Decompression	Design Security	Remote System Upgrade
FPP (×8, ×16, ×32)	✓ *(1)*	✓ *(1)*	—
AS (×1, ×4)	✓	✓	✓
PS	✓	✓	—
JTAG	—	—	—

Note to Table 9–1:

(1) In these configuration schemes, the host system must accommodate a different DCLK-to-DATA[] ratio. For more information, refer to "Fast Passive Parallel Configuration" on page 9–9.

Power-On Reset Circuit and Configuration Pins Power Supply

The following sections describe the power-on reset (POR) circuit and the power supply for the configuration pins.

POR Delay Specification

POR delay is defined as the delay between the time when all the power supplies monitored by the POR circuitry reach the minimum recommended operating voltage to the time when the nSTATUS is released high and your device is ready to begin configuration.

For more information about the POR delay, refer to the *Hot Socketing and Power-On Reset in Stratix V Devices* chapter.

Table 9–2 lists the fast and standard POR delay specification.

Table 9–2. Fast and Standard POR Delay Specification *(Note 1)*

POR Delay	Minimum	Maximum
Fast	4 ms	12 ms
Standard	100 ms	300 ms

Note to Table 9–2:

(1) You can select the POR delay based on the MSEL settings as described in Table 9–4 on page 9–7.

Power-On Reset Circuit

The POR circuit keeps the entire system in reset mode until the power supply voltage levels have stabilized on power-up. After power-up, the device does not release nSTATUS until all the power supplies monitored by the POR circuitry are above the device's POR trip point. On power down, brown-out occurs if any of the power supplies monitored by the POR circuitry drops below the threshold level of the hot-socket circuitry.

For more information about which power supplies are monitored by the POR circuitry, refer to the *Hot Socketing and Power-On Reset in Stratix V Devices* chapter.

V_{CCPGM} Pin

Stratix V devices have a power supply, V_{CCPGM}, for all dedicated configuration pins and dual-purpose pins. The supported configuration voltages are 1.8, 2.5, and 3.0 V.

Use the V_{CCPGM} pin to power all dedicated configuration inputs, dedicated configuration outputs, dedicated configuration bidirectional pins, and the dual-purpose pins that you use for configuration. The configuration input buffers do not have to share power lines with the regular I/O buffer in Stratix V devices.

The operating voltage for the configuration input pin is independent of the I/O banks power supply, V_{CCIO}, during configuration. Therefore, Stratix V devices do not require configuration voltage constraints on V_{CCIO}.

For more information about the configuration pins connections, refer to the *Stratix V Device Family Pin Connection Guidelines*.

V_{CCPD} Pin

Stratix V devices have a dedicated programming power supply, V_{CCPD}, which must be connected to 3.0 V or 2.5 V to power the I/O pre-drivers and JTAG I/O pins (TCK, TMS, TDI, TDO, and TRST).

V_{CCPD} must be greater than or equal to V_{CCIO}. If V_{CCIO} is set to 3.0 V, V_{CCPD} must be powered up to 3.0 V. If the V_{CCIO} of the bank is set to 2.5 V or lower, V_{CCPD} must be powered up to 2.5 V. This applies for all the banks containing the VCCPD and VCCIO pins.

For more information about the configuration pins power supply, refer to "Device Configuration Pins" on page 9–38.

Configuration Sequence

The following sections describe the general configuration process for the FPP, AS, and PS schemes.

Power Up

To begin the configuration process, you must fully power-up all the power supplies monitored by the POR circuitry to the appropriate voltage levels. The power supplies must ramp-up monotonically within the specified ramp-up time to ensure successful configuration.

All power supplies including the V_{CCPGM} and V_{CCPD} must ramp-up from 0 V to the desired voltage level within the ramp-up time specification. If these supplies are not ramped up within this specified time, your Stratix V device will not configure successfully. If your system cannot ramp-up the power supplies within the specified ramp-up time specification, you must hold nCONFIG low until all the power supplies are stable.

For more information about the ramp-up time specification, refer to the *Hot Socketing and Power-On Reset in Stratix V Devices* chapter.

Reset

After power-up, the Stratix V device goes through a POR. The POR delay depends on the MSEL settings. During POR, the device resets, holds nSTATUS low, clears the configuration RAM bits, and tri-states all user I/O pins. After the device successfully exits POR, all user I/O pins remain tri-stated until the device is configured.

While nCONFIG is low, the device is in reset. When the device comes out of reset, nCONFIG must be at a logic-high level in order for the device to release the open-drain nSTATUS pin. After nSTATUS is released, it is pulled high by a pull-up resistor and the device is ready to receive configuration data. Before and during configuration, all user I/O pins are tri-stated. If nIO_pullup is driven low during power up and configuration, the user I/O pins and dual-purpose I/O pins have weak pull-up resistors, which are on after the device exits POR, before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

For more information about the POR delay specification, refer to "POR Delay Specification" on page 9–2.

Configuration

Both nCONFIG and nSTATUS must be deasserted at a logic-high level in order for the configuration stage to begin. For the FPP and PS configuration schemes, the device receives configuration data on its DATA pins and the clock source on the DCLK pin. Configuration data is latched into the Stratix V device on the rising edge of DCLK. For the AS configuration scheme, the device receives configuration data on its AS_DATA[] pins and drives the clock source on the DCLK pin. Configuration data is latched into the Stratix V device on the falling edge of DCLK.

After the Stratix V device has received all the configuration data successfully, it releases the CONF_DONE pin, which is pulled high by a pull-up resistor. A low-to-high transition on CONF_DONE indicates configuration has completed and initialization of the device can begin. For the FPP and PS schemes, DCLK must not be left floating at the end of configuration. You must drive them either high or low, whichever is convenient on your board.

☞ For the FPP and PS schemes, there is no maximum DCLK period, which means you can stop the configuration by holding the DCLK low for an indefinite amount of time. To resume configuration, the external host must provide data on the DATA[] pins prior to sending the first DCLK rising edge.

Configuration Error

If the **Auto-restart configuration after error** option (available in the Quartus® II software from the **General** panel of the **Device and Pin Options** dialog box) is turned on, the Stratix V device releases the nSTATUS pin high after the specified time indicated by tSTATUS and retries the configuration. If this option is turned off or if you are using a PS or FPP scheme with an external controller, the system must monitor the nSTATUS for errors and sends a low-to-high signal on nCONFIG for specified t_{CFG} time to restart the configuration.

Initialization

In Stratix V devices, initialization begins after the CONF_DONE goes high. For the FPP and PS configuration schemes, two DCLK falling edges are required after the last configuration byte is sent to the Stratix V device to begin the initialization of the device for both uncompressed and compressed configuration data.

The initialization clock source is from the internal oscillator, CLKUSR, or DCLK pin. By default, the internal oscillator is the clock source for initialization. If you use the internal oscillator, the Stratix V device provides itself with enough clock cycles for proper initialization.

Table 9–3 lists the initialization clock source option, the applicable configuration schemes, and the maximum frequency.

Table 9–3. Initialization Clock Source Option and the Maximum Frequency

Initialization Clock Source	Configuration Schemes	Maximum Frequency	Minimum Number of Clock Cycles *(1)*
Internal Oscillator	AS, PS, FPP	12.5 MHz	17,408 *(3)*
CLKUSR	AS, PS, FPP *(2)*	125 MHz	

Notes to Table 9–3:

(1) The minimum number of clock cycles required for device initialization.

(2) To enable CLKUSR as the initialization clock source, turn on the **Enable user-supplied start-up clock (CLKUSR)** option in the Quartus II software from the **General** panel of the **Device and Pin Options** dialog box.

(3) The number is still preliminary.

☞ If you use the optional CLKUSR pin as the initialization clock source and nCONFIG is pulled low to restart configuration during device initialization, ensure that CLKUSR or DCLK continues toggling until nSTATUS goes low and goes high again.

CLKUSR provides you with the flexibility to synchronize initialization of multiple devices or to delay initialization. Supplying a clock on the CLKUSR pin during initialization does not affect configuration. After CONF_DONE goes high, CLKUSR or DCLK is enabled after the time specified by t_{CD2CU}. When this time period elapses, Stratix V devices require a minimum number of clock cycles to initialize properly and enter user mode as specified by the t_{CD2UMC} parameter.

User Mode

The Stratix V device enters user mode when the initialization is complete. You can monitor the end of the initialization stage by enabling the optional INIT_DONE pin. If enabled, the low-to-high transition of INIT_DONE indicates the device has completed initialization and entered user mode. In this mode, your design is executed. The user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

At any time during the configuration stage or user mode operation, you can initiate a reconfiguration by setting a low pulse on the nCONFIG pin. The pulse must meet the minimum t_{CFG} low-pulse width. When nCONFIG is pulled low, the nSTATUS and CONF_DONE pins are pulled low and all I/O pins are tri-stated. Configuration begins when the nCONFIG and nSTATUS pins return to a logic-high level.

Configuration Schemes

The following sections describe configuration schemes for Stratix V devices.

MSEL Pin Settings

Select the configuration scheme by driving the Stratix V device MSEL pins either high or low (refer to Table 9–4). The MSEL input buffers are powered by the V_{CCPGM} power supply. Altera recommends hardwiring the MSEL pins to V_{CCPGM} or GND. During POR and during reconfiguration, the MSEL pins must be at the LVTTL V_{IL} and V_{IH} levels to be considered logic low and logic high, respectively.

To avoid problems with detecting an incorrect configuration scheme, hardwire the MSEL pins to V_{CCPGM} or GND without pull-up or pull-down resistors. Do not drive the MSEL pins with a microprocessor or another device.

Table 9–4 lists the configuration schemes for Stratix V devices.

Table 9–4. Configuration Schemes for Stratix V Devices (Part 1 of 2)

Configuration Scheme	Decompression Feature	Design Security Feature	Configuration Voltage Standard (V) *(2)*	POR Delay *(5)*	MSEL[4..0]
FPP ×8	Disabled	Disabled	1.8/2.5/3.0	Fast	10100
				Standard	11000
	Disabled	Enabled	1.8/2.5/3.0	Fast	10101
				Standard	11001
	Enabled	Optional *(1)*	1.8/2.5/3.0	Fast	10110
				Standard	11010
FPP ×16	Disabled	Disabled	1.8/2.5/3.0	Fast	00000
				Standard	00100
	Disabled	Enabled	1.8/2.5/3.0	Fast	00001
				Standard	00101
	Enabled	Optional *(1)*	1.8/2.5/3.0	Fast	00010
				Standard	00110
FPP ×32	Disabled	Disabled	1.8/2.5/3.0	Fast	01000
				Standard	01100
	Disabled	Enabled	1.8/2.5/3.0	Fast	01001
				Standard	01101
	Enabled	Optional *(1)*	1.8/2.5/3.0	Fast	01010
				Standard	01110
PS	Optional *(1)*	Optional *(1)*	1.8/2.5/3.0	Fast	10000
				Standard	10001
AS (×1, ×4) *(3)*	Optional *(1)*	Optional *(1)*	3.0	Fast	10010
				Standard	10011

Table 9–4. Configuration Schemes for Stratix V Devices (Part 2 of 2)

Configuration Scheme	Decompression Feature	Design Security Feature	Configuration Voltage Standard (V) *(2)*	POR Delay *(5)*	MSEL[4..0]
JTAG-based configuration *(4)*	Disabled	Disabled	—	—	*(6)*

Notes to Table 9–4:

(1) You can select to enable or disable this feature.

(2) The configuration voltage standard applied to the V_{CCPGM} power supply that powers all the configuration pins during configuration.

(3) The AS configuration scheme supports the remote system upgrade feature. For more information about the remote system upgrade feature, refer to "Remote System Upgrades" on page 9–44.

(4) JTAG-based configuration takes precedence over other configuration schemes. This means the MSEL pin settings are ignored. JTAG-based configuration does not support the design security or decompression features.

(5) For POR delay specification, refer to "POR Delay Specification" on page 9–2.

(6) Do not leave the MSEL pins floating. Connect them to V_{CCPGM} or GND. This pin supports the non-JTAG configuration scheme used in production. If you only use the JTAG configuration, Altera recommends connecting the MSEL pins to GND.

Raw Binary File Size

For the POR delay specification, refer to "POR Delay Specification" on page 9–2.

Table 9–5 lists the uncompressed raw binary file (**.rbf**) sizes for Stratix V devices.

Table 9–5. Uncompressed .rbf Sizes for Stratix V Devices *(Note 1)*

Family	Device	Configuration .rbf Size (bits)
Stratix V GX	5SGXA3	139,255,840
	5SGXA4	139,255,840
	5SGXA5	266,599,584
	5SGXA7	266,599,584
	5SGXA9	387,394,048
	5SGXAB	387,394,048
	5SGXB5	266,798,896
	5SGXB6	266,798,896
Stratix V GT	5SGTC5	266,035,472
	5SGTC7	266,035,472
Stratix V GS	5SGSD2	93,080,448
	5SGSD3	93,080,448
	5SGSD4	209,935,224
	5SGSD5	209,935,224
	5SGSD6	266,798,896
	5SGSD8	266,798,896
Stratix V E	5SEE9	387,394,048
	5SEEB	387,394,048

Note to Table 9–5:

(1) These values are preliminary.

Use the data in Table 9–5 to estimate the file size before design compilation. Different configuration file formats, such as a hexadecimal (**.hex**) or tabular text file (**.ttf**) format, have different file sizes. For the different types of configuration file and file sizes, refer to the Quartus II software. However, for a specific version of the Quartus II software, any design targeted for the same device has the same uncompressed configuration file size. If you are using compression, the file size can vary after each compilation because the compression ratio depends on your design.

For more information about setting device configuration options or creating configuration files, refer to the *Device Configuration Options* and *Configuration File Formats* chapters in volume 2 of the *Configuration Handbook*.

Fast Passive Parallel Configuration

The FPP configuration using an external host provides the fastest method to configure Stratix V devices. FPP is supported in multiple data widths—8-bits, 16-bits, and 32-bits. You can perform a FPP configuration of Stratix V devices using an external host such as a MAX II device or microprocessor. The external host controls the transfer of configuration data from a storage device, such as flash memory, to the target Stratix V device. You can store configuration data in **.rbf**, **.hex**, or **.ttf** formats. Therefore, the design that controls the configuration stages, such as fetching the data from flash memory and sending it to the device, must be stored in the MAX II device or microprocessor.

The Parallel Flash Loader (PFL) feature in MAX II devices provides an efficient method to program CFI flash memory devices through the JTAG interface. PFL also acts as a controller to read configuration data from the flash memory device and configures the Stratix V device. PFL supports both the PS and FPP configuration schemes.

For more information about the PFL, refer to *Parallel Flash Loader Megafunction User Guide*.

Two DCLK falling edges are required after CONF_DONE goes high to begin the initialization of the device for both uncompressed and compressed configuration data in a FPP configuration.

DCLK-to-DATA[] Ratio for FPP configuration

FPP configuration requires a different DCLK-to-DATA[] ratio when you enable the design security, decompression, or both features. Table 9–6 lists the DCLK-to-DATA[] ratio for each combination.

Table 9–6. DCLK-to-DATA[] Ratio *(Note 1)* (Part 1 of 2)

Configuration Scheme	Decompression	Design Security	DCLK-to-DATA[] Ratio
FPP ×8	Disabled	Disabled	1
	Disabled	Enabled	1
	Enabled	Disabled	2
	Enabled	Enabled	2

Table 9–6. DCLK-to-DATA[] Ratio *(Note 1)* (Part 2 of 2)

Configuration Scheme	Decompression	Design Security	DCLK-to-DATA[] Ratio
FPP ×16	Disabled	Disabled	1
	Disabled	Enabled	2
	Enabled	Disabled	4
	Enabled	Enabled	4
FPP ×32	Disabled	Disabled	1
	Disabled	Enabled	4
	Enabled	Disabled	8
	Enabled	Enabled	8

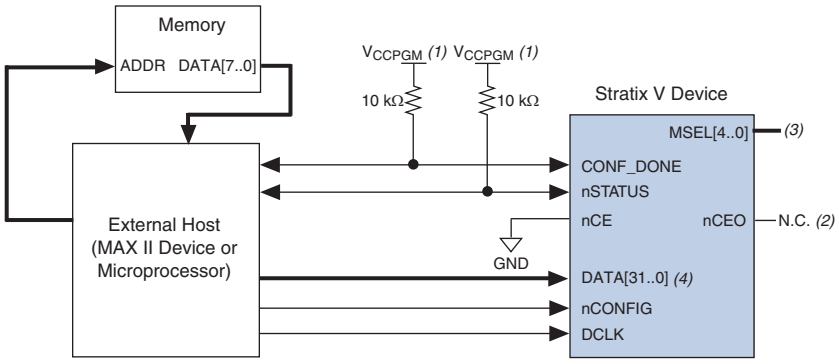
Note to Table 9–6:

(1) Depending on the DCLK-to-DATA[] ratio, the host must send a DCLK frequency that is r times the data rate in bytes per second (Bps), or words per second (Wps). For example, in FPP ×16 when the DCLK-to-DATA[] ratio is 2, the DCLK frequency must be 2 times the data rate in Wps. Stratix V devices use the additional clock cycles to decrypt and decompress the configuration data.

If the DCLK-to-DATA[] ratio is greater than 1, at the end of configuration, you can only stop the DCLK (DCLK-to-DATA[] ratio – 1) clock cycles after the last data is latched into the Stratix V device.

Figure 9–1 shows the configuration interface connections between the Stratix V device and a MAX II device for single device configuration.

Figure 9–1. Single Device FPP Configuration Using an External Host



Notes to Figure 9–1:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Stratix V device. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends powering up all configuration system I/Os with V_{CCPGM}.

(2) You can leave the nCEO pin unconnected or use it as a user I/O pin when it does not feed another device's nCE pin.

(3) The MSEL pin settings vary for different data width, configuration voltage standards, and POR delay. To connect MSEL, refer to Table 9–4 on page 9–7.

(4) If you use FPP ×8, use DATA[7..0]. If you use FPP ×16, use DATA[15..0].

FPP Multi-Device Configuration

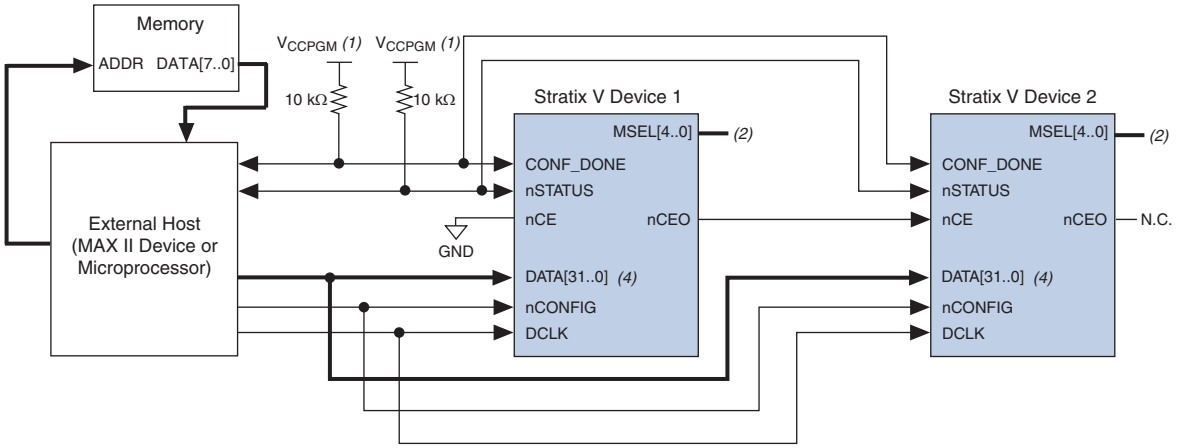
For FPP multi-device configuration, you can configure all devices with different sets of configuration data (multiple SRAM object files [**.sof**s]) or with the same configuration data (single **.sof**). In both cases, the nCONFIG, nSTATUS, DCLK, DATA[], and CONF_DONE pins are connected to every device in the chain. Ensure that the DCLK and data line are buffered for every fourth device. This ensures the signal integrity and prevents clock skew problems.

Because all device's CONF_DONE and nSTATUS pins are tied together, all devices initialize and enter user mode at the same time. If any device detects an error, configuration stops for the entire chain and you must reconfigure all devices. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

☞ For FPP multi-device configuration, all devices in the chain must have the same data width. If you are using FPP ×32, all devices in the chain must use FPP ×32 configuration scheme. If you are using FPP ×8, you can use the Stratix V device with other FPGA devices that support FPP ×8.

Figure 9–2 shows how to configure multiple devices using a MAX II device when both devices receive a different set of configuration data (multiple **.sof**s).

Figure 9–2. Multi-Device FPP Configuration Using an External Host When Both Devices Receive a Different Set of Configuration Data



Notes to Figure 9–2:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Stratix V device. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends powering up all configuration system I/Os with V_{CCPGM}.

(2) You can leave the nCEO pin unconnected or use it as a user I/O pin when it does not feed another device's nCE pin.

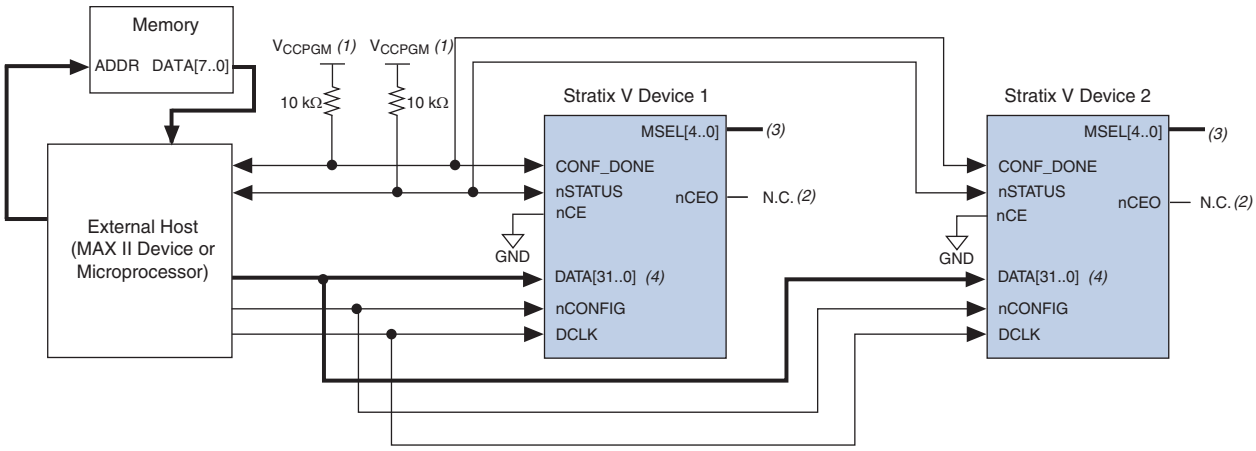
(3) The MSEL pin settings vary for different data width and POR delay. To connect MSEL, refer to Table 9–4 on page 9–7.

(4) If you use FPP ×8, use DATA[7..0]. If you use FPP ×16, use DATA[15..0]. All devices in the chain must have the same data width.

In Figure 9–2, after the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration in one clock cycle; therefore, the transfer of data to the second device is transparent to the MAX II device or microprocessor.

Figure 9–3 shows the FPP configuration setup for multiple devices when both Stratix V devices receive the same configuration data (single **.sof**).

Figure 9–3. Multiple Device FPP Configuration Using an External Host When Both Devices Receive the Same Data



Notes to Figure 9–3:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Stratix V device. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends powering up all configuration system I/Os with V_{CCPGM}.

(2) You can leave the `nCEO` pin unconnected or use it as a user I/O pin when it does not feed another device's `nCE` pin.

(3) The `MSEL` pin settings vary for different data width and POR delay. To connect `MSEL`, refer to Table 9–4 on page 9–7.

(4) If you use FPP ×8, use `DATA[7..0]`. If you use FPP ×16, use `DATA[15..0]`. All devices in the chain must have the same data width.

In Figure 9–3, because both nCE pins are tied to GND, both devices in the chain begin and complete the configuration and enter user mode at the same time.

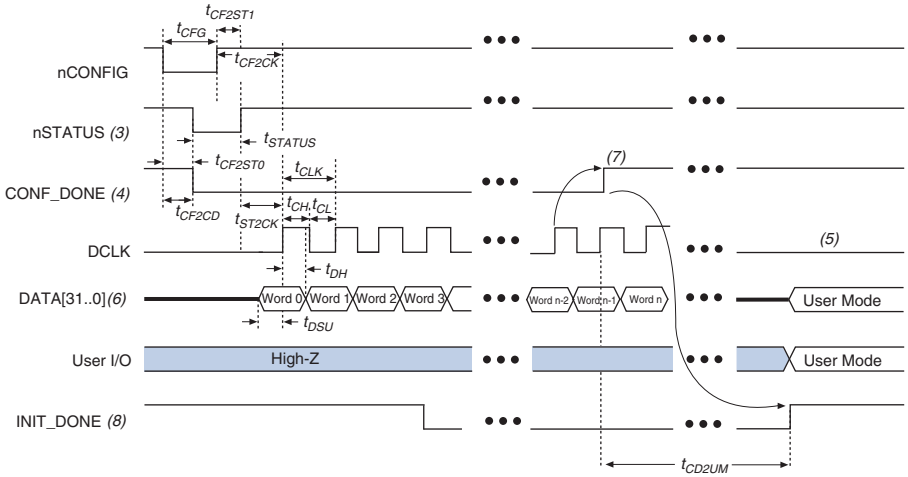
To configure FPP multi-device with a single **.sof**, all Stratix V devices in the chain must be in the same package and density.

FPP Configuration Timing

Figure 9–4 shows the timing waveform for FPP configuration when using a MAX II device as an external host. This waveform shows timing when the DCLK-to-DATA[] ratio is 1.

When you enable the decompression or design security feature, the DCLK-to-DATA[] ratio varies for FPP ×8, FPP ×16, and FPP ×32. For the respective DCLK-to-DATA[] ratio, refer to Table 9–6 on page 9–9.

Figure 9–4. FPP Configuration Timing Waveform When the DCLK-to-DATA[] Ratio is 1 ***(Note 1), (2)***



Notes to Figure 9–4:

(1) Use this timing waveform when the DCLK-to-DATA[] ratio is 1.

(2) The beginning of this waveform shows the device in user mode. In user mode, nCONFIG, nSTATUS, and CONF_DONE are at logic-high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.

(3) After power-up, the Stratix V device holds nSTATUS low for the time of the POR delay.

(4) After power-up, before and during configuration, CONF_DONE is low.

(5) Do not leave DCLK floating after configuration. You can drive it high or low, whichever is more convenient.

(6) For FPP ×16, use DATA[15..0]. For FPP ×8, use DATA[7..0]. DATA[31..0] are available as a user I/O pin after configuration. The state of this pin depends on the dual-purpose pin settings.

(7) To ensure a successful configuration, send the entire configuration data to the Stratix V device. CONF_DONE is released high when the Stratix V device receives all the configuration data successfully. After CONF_DONE goes high, send two additional falling edges on DCLK to begin initialization and enter user mode.

(8) After the option bit to enable the INIT_DONE pin is configured into the device, the INIT_DONE goes low.

Table 9–7 lists the timing parameters for Stratix V devices for FPP configuration when the DCLK-to-DATA[] ratio is 1.

Table 9–7. FPP Timing Parameters for Stratix V Devices *(Note 1)*, *(2)*

Symbol	Parameter	Minimum	Maximum	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low	—	600	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low	—	600	ns
t_{CFG}	nCONFIG low pulse width	2	—	µs
t_{STATUS}	nSTATUS low pulse width	268	1,506 *(3)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high	—	1,506 *(4)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	1,506	—	µs
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2	—	µs
t_{DSU}	DATA[] setup time before rising edge on DCLK	5.5	—	ns
t_{DH}	DATA[] hold time after rising edge on DCLK	0	—	ns
t_{CH}	DCLK high time	0.45 × 1/f_{MAX}	—	s
t_{CL}	DCLK low time	0.45 × 1/f_{MAX}	—	s
t_{CLK}	DCLK period	1/f_{MAX}	—	s
f_{MAX}	DCLK frequency (FPP ×8/×16)	—	125	MHz
	DCLK frequency (FPP ×32)	—	100	MHz
t_R	Input rise time	—	40	ns
t_F	Input fall time	—	40	ns
t_{CD2UM}	CONF_DONE high to user mode *(5)*	175	437	µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period	—	—
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	t_{CD2CU} + (17,408 × CLKUSR period) *(6)*	—	—

Notes to Table 9–7:

(1) This information is preliminary.

(2) Use these timing parameters when the decompression and design security features are disabled.

(3) This value is applicable if you do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

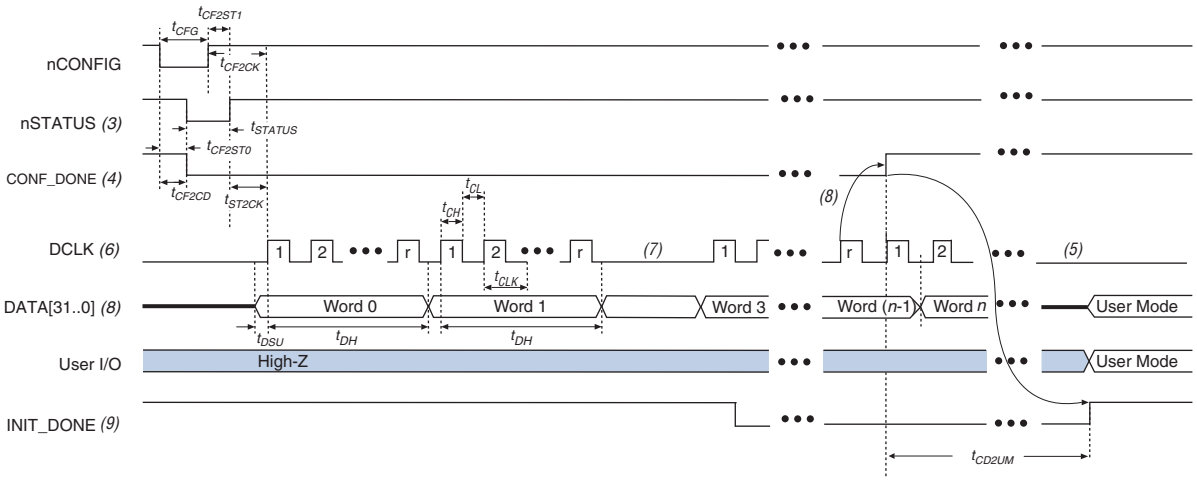
(4) This value is applicable if you do not delay configuration by externally holding the nSTATUS low.

(5) The minimum and maximum numbers apply only if you chose the internal oscillator as the clock source for initializing the device.

(6) To enable the CLKUSR pin as the initialization clock source and to obtain the maximum frequency specification on these pins, refer to "Initialization" on page 9–5.

Figure 9–5 shows the timing waveform for FPP configuration when using a MAX II device or microprocessor as an external host. This waveform shows timing when the DCLK-to-DATA[] ratio is more than 1.

Figure 9–5. FPP Configuration Timing Waveform When the DCLK-to-DATA[] Ratio is >1 ***(Note 1), (2)***



Notes to Figure 9–5:

(1) Use this timing waveform and parameters when the DCLK-to-DATA[] ratio is >1. To find out the DCLK-to-DATA[] ratio for your system, refer Table 9–6 on page 9–9.

(2) The beginning of this waveform shows the device in user mode. In user mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.

(3) After power-up, the Stratix V device holds nSTATUS low for the time as specified by the POR delay.

(4) After power-up, before and during configuration, CONF_DONE is low.

(5) Do not leave DCLK floating after configuration. You can drive it high or low, whichever is more convenient.

(6) "r" denotes the DCLK-to-DATA[] ratio. For the DCLK-to-DATA[] ratio based on the decompression and the design security feature enable settings, refer to Table 9–6 on page 9–9.

(7) If needed, pause DCLK by holding it low. When DCLK restarts, the external host must provide data on the DATA[31..0] pins prior to sending the first DCLK rising edge.

(8) To ensure a successful configuration, send the entire configuration data to the Stratix V device. CONF_DONE is released high after the Stratix V device receives all the configuration data successfully. After CONF_DONE goes high, send two additional falling edges on DCLK to begin initialization and enter user mode.

(9) After the option bit to enable the INIT_DONE pin is configured into the device, the INIT_DONE goes low.

Table 9–8 lists the timing parameters for Stratix V devices for FPP configuration when the DCLK-to-DATA[] ratio is more than 1.

Table 9–8. FPP Timing Parameters for Stratix V Devices When the DCLK-to-DATA[] Ratio is >1 ***(Note 1), (2)*** **(Part 1 of 2)**

Symbol	Parameter	Minimum	Maximum	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low	—	600	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low	—	600	ns
t_{CFG}	nCONFIG low pulse width	2	—	µs
t_{STATUS}	nSTATUS low pulse width	268	1,506 *(3)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high	—	1,506 *(3)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	1,506	—	µs
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2	—	µs
t_{DSU}	DATA[] setup time before rising edge on DCLK	5.5	—	ns

Table 9–8. FPP Timing Parameters for Stratix V Devices When the DCLK-to-DATA[] Ratio is >1 *(Note 1)*, *(2)* (Part 2 of 2)

Symbol	Parameter	Minimum	Maximum	Units
t_{DH}	DATA[] hold time after rising edge on DCLK	3 × 1/f_{DCLK} *(6)*	—	s
t_{CH}	DCLK high time	0.45 × 1/f_{MAX}	—	s
t_{CL}	DCLK low time	0.45 × 1/f_{MAX}	—	s
t_{CLK}	DCLK period	1/f_{MAX}	—	s
f_{MAX}	DCLK frequency (FPP ×8/×16)	—	125	MHz
	DCLK frequency (FPP ×32)	—	100	MHz
t_R	Input rise time	—	40	ns
t_F	Input fall time	—	40	ns
t_{CD2UM}	CONF_DONE high to user mode *(4)*	175	437	µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period	—	—
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	t_{CD2CU} + (17,408 × CLKUSR period) *(5)*	—	—

Notes to Table 9–8:

(1) This information is preliminary.

(2) Use these timing parameters when you use the decompression and design security features.

(3) You can obtain this value if you do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(4) The minimum and maximum numbers apply only if you use the internal oscillator as the clock source for initializing the device.

(5) To enable the CLKUSR pin as the initialization clock source and to obtain the maximum frequency specification on these pins, refer to "Initialization" on page 9–5.

(6) f_{DCLK} is the DCLK frequency the system is operating.

Active Serial Configuration (Serial Configuration Devices)

The AS configuration scheme is supported in 1-bit data wide (AS ×1 mode) or 4-bit data wide (AS ×4 mode). In the AS ×1 mode, the Stratix V devices are configured using a serial configuration device (EPCS). In the AS ×4 mode, a quad-serial configuration device (EPCQ) configures the Stratix V devices. The AS ×4 mode provides four times faster configuration time than the AS ×1 mode.

EPCS and EPCQ are low-cost devices with non-volatile memory that feature a simple four-pin interface or six-pin interface, respectively, and a small form factor. These features make the AS configuration scheme an ideal low-cost configuration solution.

If you wish to gain control of the EPCS pins, hold the nCONFIG pin low and pull the nCE pin high. This causes the device to reset and tri-state the AS configuration pins.

For more information about EPCS and EPCQ devices, refer to the volume 2 of the *Configuration Handbook*.

AS mode supports DCLK frequency up to 100 MHz. You can choose the CLKUSR or internal oscillator as the configuration clock source that drives DCLK. If you use the internal oscillator as the configuration clock source, you can choose a 12.5, 25, 50, or 100 MHz clock from the **Configuration** panel in the **Device and Pins Option** settings.

Table 9–9 lists the DCLK frequency specification in the AS configuration scheme.

Table 9–9. DCLK Frequency Specification in the AS Configuration Scheme ***(Note 1)***, ***(2)***

Minimum	Typical	Maximum	Unit
5.3	7.9	12.5	MHz
10.6	15.7	25.0	MHz
21.3	31.4	50.0	MHz
42.6	62.9	100.0	MHz

Notes to Table 9–9:

(1) This information is preliminary.

(2) This applies to the DCLK frequency specification when using the internal oscillator as the configuration clock source.

You can choose the internal oscillator or CLKUSR as the DCLK clock source by selecting the option under **Device and Pins Option** settings, **Configuration** panel in the Quartus II software. This sets specific option in the programming file. By default, in AS scheme, Stratix V devices power-up and begin configuration with 12.5 MHz internal oscillator as the DCLK clock source. After reading the option bits from the programming file, Stratix V devices continue using the internal oscillator at 12.5 MHz frequency, switch to a higher internal oscillator clock frequency, or switch to the CLKUSR pin.

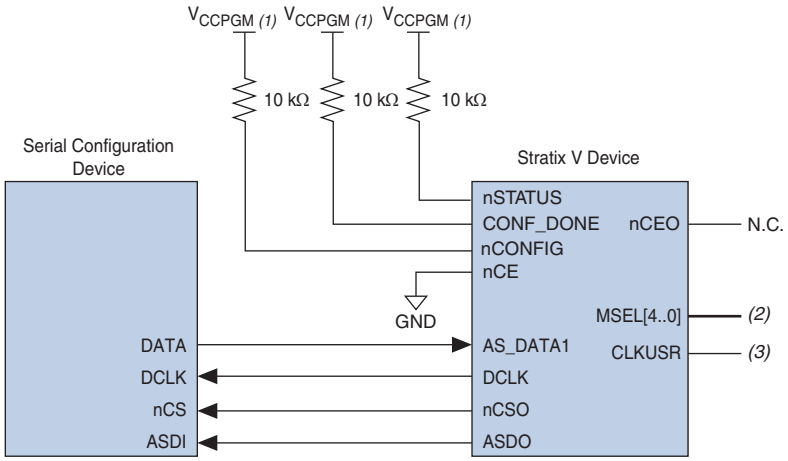
If you choose CLKUSR as configuration clock source, the maximum frequency allowed is 100 MHz.

During device configuration, Stratix V devices read the configuration data using the serial interface, decompress the data if necessary, and configure their SRAM cells. In the AS scheme, the Stratix V device controls the configuration interface. In the PS scheme, the external host (a MAX II device or microprocessor) controls the interface.

You can select between the AS ×1 and AS ×4 settings by selecting the option under **Device and Pins Option** settings, **Configuration** panel in the Quartus II software. This sets a specific option bit in the programming file. By default, in the AS scheme, Stratix V devices power-up and begin configuration as an AS ×1 mode. Upon reading the option bits from the programming file, Stratix V devices either stay as an AS ×1 mode or switch to an AS ×4 mode for the rest of the configuration.

Figure 9–6 shows the single-device configuration setup for an AS ×1 mode.

Figure 9–6. Single Device AS ×1 Mode Configuration



Notes to Figure 9–6:

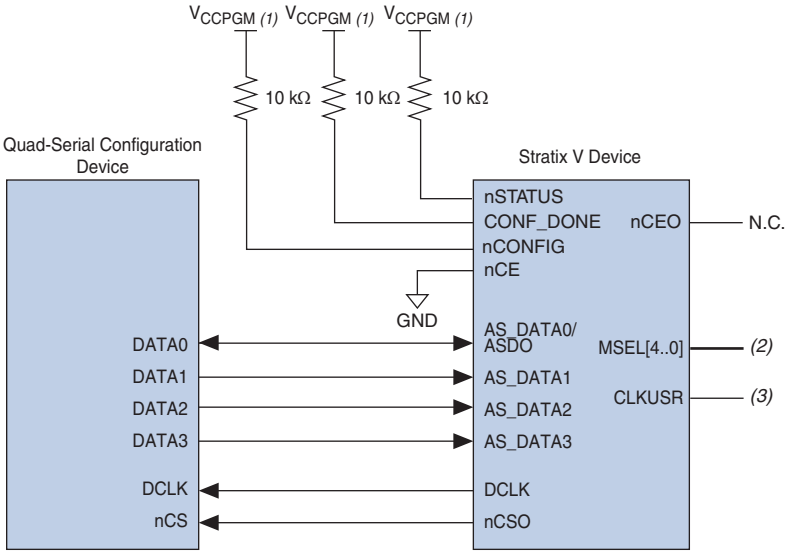
(1) Connect the pull-up resistors to V_{CCPGM} at 3.0-V supply.

(2) The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect MSEL, refer to Table 9–4 on page 9–7.

(3) You can use the CLKUSR pin to supply the external clock source to drive the DCLK during configuration. The maximum frequency specification is 100 MHz.

Figure 9–7 shows the single-device configuration setup for an AS ×4 mode.

Figure 9–7. Single Device AS ×4 Mode Configuration



Notes to Figure 9–7:

(1) Connect the pull-up resistors to V_{CCPGM} at 3.0-V supply.

(2) The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect MSEL, refer to Table 9–4 on page 9–7.

(3) You can use the CLKUSR pin to supply the external clock source to drive the DCLK during configuration. The maximum frequency specification is 100 MHz.

The serial clock (DCLK) generated by the Stratix V device controls the entire configuration cycle and provides timing for the serial interface. Stratix V devices use an internal oscillator or external clock (CLKUSR) as the configuration clock source (DCLK). In the AS configuration scheme, Stratix V devices drive out control signals on the falling edge of DCLK and latch in the data on the following falling edge of DCLK.

During configuration, Stratix V devices enable the EPCS or EPCQ by driving the nCSO output pin low, which connects to the chip select (nCS) pin of the EPCS or EPCQ. Stratix V devices use the serial clock (DCLK) and serial data output (ASDO) pins to send operation commands and read address signals to the EPCS or EPCQ. The EPCS or EPCQ provides data on its serial data output (DATA[]) pin, which connects to the AS_DATA[] input of the Stratix V devices.

AS Multi-Device Configuration

For the AS multi-device configuration scheme, you can configure all devices with different sets of configuration data (different **.sof**) or with the same configuration data (same **.sof**). In both cases, the nCONFIG, nSTATUS, DCLK, and data line (AS_DATA1 on the master device and DATA0 on the slave device) and CONF_DONE pins are connected to every device in the chain. Ensure that the DCLK and data line are buffered for every fourth device.

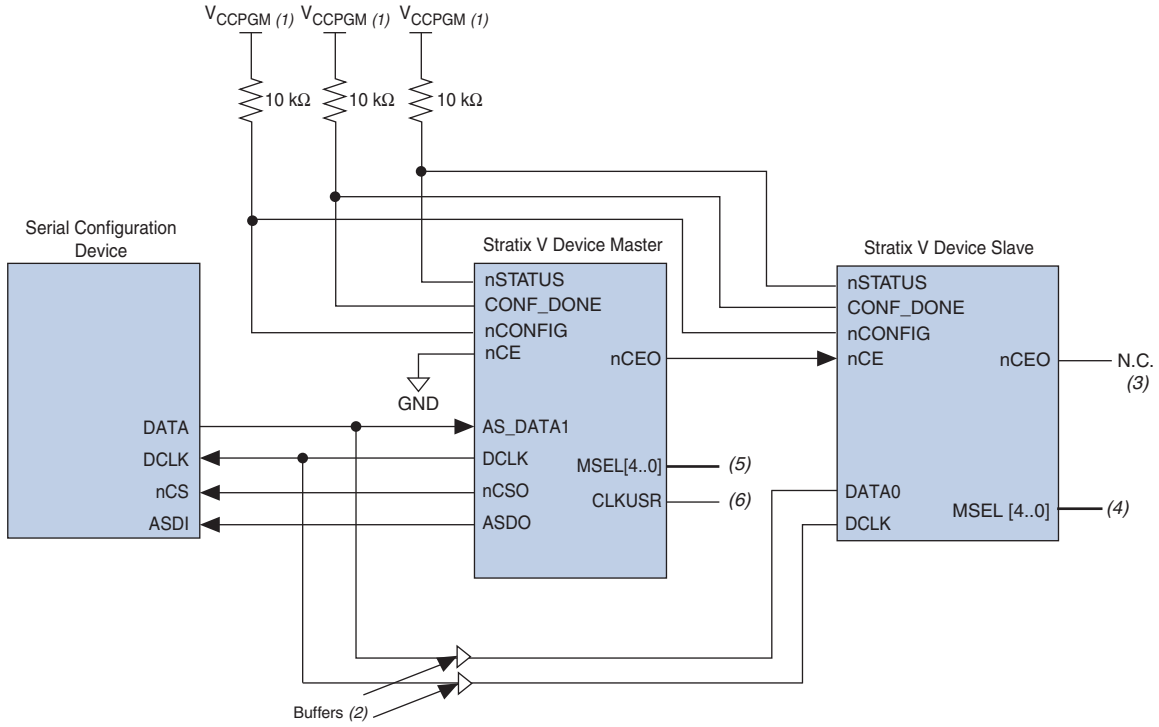
The AS configuration scheme supports multi-device in AS ×1 mode. The AS ×4 mode does not support multi-device configuration setup.

In the AS multi-device configuration, the nSTATUS, nCONFIG, and CONF_DONE pins are tied together. Therefore, all devices initialize and enter user mode at the same time. If any device detects an error, configuration stops for the entire chain and you must reconfigure all devices. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

In this configuration scheme, the first Stratix V device in the chain is the configuration master and controls the configuration of the entire chain. You must connect its MSEL pins to select the AS configuration scheme. The remaining Stratix V devices are configuration slaves. You must connect their MSEL pins to select the PS configuration scheme. Any other Altera® device that supports a PS configuration can also be part of the chain as the configuration slave.

Figure 9–8 shows the multi-device configuration setup for AS ×1 mode when both devices in the chain receive different sets of configuration data (multiple **.sof**s).

Figure 9–8. AS Multi-Device Configuration When Both Devices in the Chain Receive Different Sets of Configuration Data *(Note 1)*, *(2)*



Notes to Figure 9–8:

(1) Connect the pull-up resistors to V_{CCPGM} at a 3.0-V supply.

(2) Connect the repeater buffers between the Stratix V master and slave device for AS_DATA1/DATA0 and DCLK.

(3) You can leave the nCEO pin unconnected or use it as a user I/O pin when it does not feed another device's nCE pin.

(4) For the appropriate MSEL settings based on POR delay settings, set the slave device MSEL setting to the PS scheme. Refer to Table 9–4 on page 9–7.

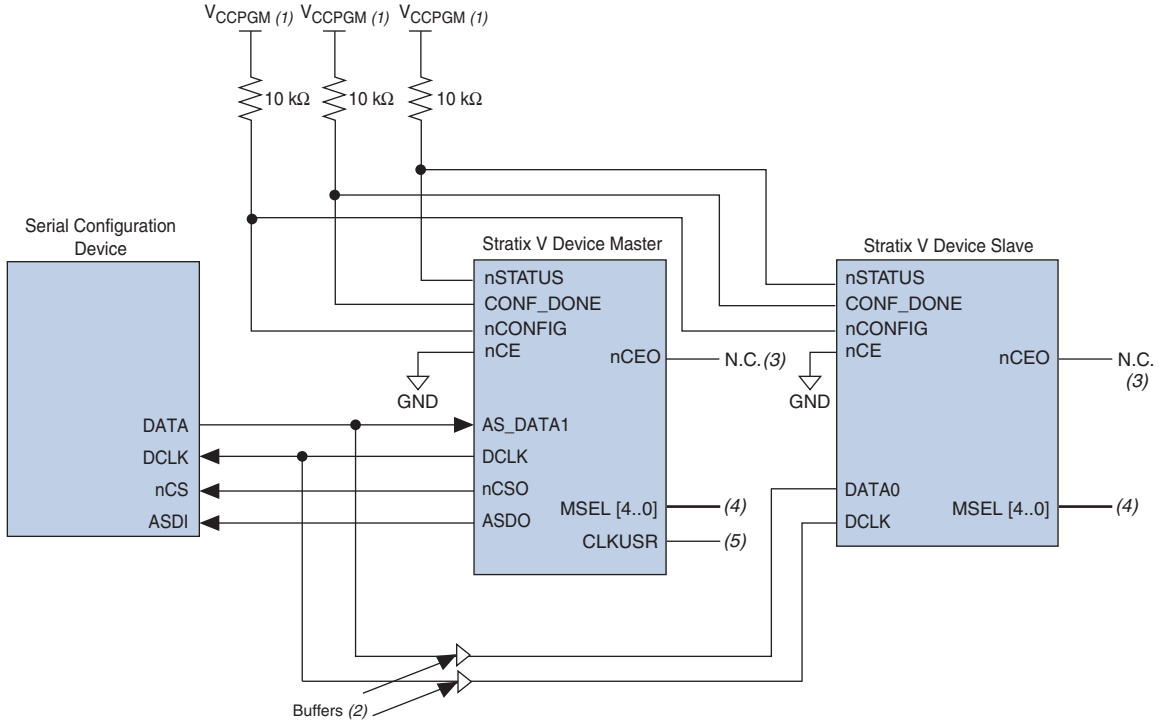
(5) The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect MSEL, refer to Table 9–4 on page 9–7.

(6) You can use the CLKUSR pin to supply the external clock source to drive the DCLK during configuration. The maximum frequency specification is 100 MHz.

In Figure 9–8, after the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration in one clock cycle; therefore, the transfer of data to the second device is transparent to the first device in the chain.

Figure 9–9 shows the multi-device configuration setup for AS ×1 mode when all devices in the chain receive the same set of configuration data (single **.sof**).

Figure 9–9. AS Multi-Device Configuration When the Devices Receive the Same Data Using a Single .sof ***(Note 1)***



Notes to Figure 9–9:

(1) Connect the pull-up resistors to V_{CCPGM} at a 3.0-V supply.

(2) Connect the repeater buffers between the Stratix V master and slave device for AS_DATA1/DATA0 and DCLK.

(3) You can leave the nCEO pin unconnected or use it as a user I/O pin when it does not feed another device's nCE pin.

(4) The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect the MSEL, refer to Table 9–4 on page 9–7.

(5) You can use the CLKUSR pin to supply the external clock source to drive the DCLK during configuration. The maximum frequency specification is 100 MHz.

AS Connection Guidelines

Table 9–10 lists the board trace length and loading between the supported EPCS or EPCQ and Stratix V devices for single- and multi-device AS configurations.

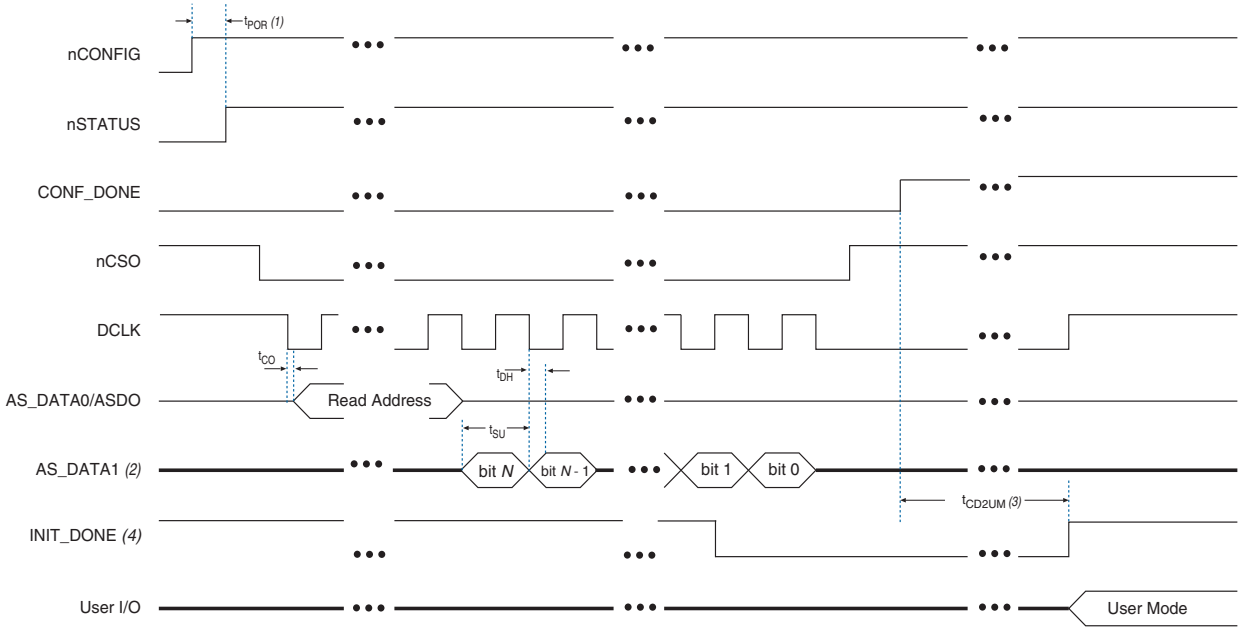
Table 9–10. Maximum Trace Length and Loading for AS ×1/×4 Configurations

Stratix V Device AS Pins	Maximum Board Trace Length from the Stratix V Device to the Serial Configuration Device for 12.5/ 25/ 50 MHz Operation (Inches)	Maximum Board Trace Length from the Stratix V Device to the Serial Configuration Device for 100 MHz Operation (Inches)	Maximum Board Load (pF)
DCLK	10	6	15
DATA[3..0]	10	6	30
nCSO	10	6	30

AS Configuration Timing

Figure 9–10 shows the timing waveform for the AS ×1 mode and AS ×4 mode configuration timing.

Figure 9–10. AS Configuration Timing



Notes to Figure 9–10:

(1) The AS scheme supports standard and fast POR delay (t_{POR}). For t_{POR} delay information, refer to "POR Delay Specification" on page 9–2.
(2) If you are using AS ×4 mode, this signal represents the AS_DATA[3..0] and EPCQ sends in 4-bits of data for each DCLK cycle.
(3) The initialization clock can be from internal oscillator or CLKUSR pin.
(4) After the option bit to enable the INIT_DONE pin is configured into the device, the INIT_DONE goes low.

Table 9–11 lists the timing parameters for AS ×1 and AS ×4 configurations in Stratix V devices.

Table 9–11. AS Timing Parameters for AS ×1 and AS ×4 Configurations in Stratix V Devices ***(Note 1)***, ***(2)***, ***(3)***

Symbol	Parameter	Minimum	Maximum	Units
t_{CO}	DCLK falling edge to AS_DATA0/ASDO output	—	4	µs
t_{SU}	Data setup time before rising edge on DCLK	1.5	—	ns
t_{H}	Data hold time after rising edge on DCLK	0	—	ns
t_{CD2UM}	CONF_DONE high to user mode *(4)*	175	437	µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period	—	—
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	t_{CD2CU} + (17,408 × CLKUSR period)	—	—

Notes to Table 9–11:

(1) This information is preliminary.

(2) The minimum and maximum numbers apply only if you choose the internal oscillator as the clock source for initializing the device.

(3) t_{CF2CD}, t_{CF2ST0}, t_{CFG}, t_{STATUS}, and t_{CF2ST1} timing parameters are identical to the timing parameters for PS mode listed in Table 9–12 on page 9–30.

(4) To enable the CLKUSR pin as the initialization clock source and to obtain the maximum frequency specification on this pin, refer to "Initialization" on page 9–5.

Estimating the Active Serial Configuration Time

The AS configuration time is dominated by the time it takes to transfer data from the EPCS to the Stratix V device. This serial interface is clocked by the Stratix V DCLK.

You can estimate the minimum AS ×1 mode configuration time by using the following equation:

.rbf Size × (minimum DCLK period / 1 bit per DCLK cycle) = estimated minimum configuration time.

You can estimate the minimum AS ×4 mode configuration time by using the following equation:

.rbf Size × (minimum DCLK period / 4 bits per DCLK cycle) = estimated minimum configuration time.

Enabling compression reduces the amount of configuration data that is transmitted to the Stratix V device, which also reduces the configuration time. Your configuration time is reduced based on the compression ratio. The compression ratio varies based on the design.

Programming EPCS and EPCQ

EPCS and EPCQ are non-volatile, flash-memory-based devices. You can program these devices in-system using a USB-Blaster™, EthernetBlaster, or ByteBlaster™ II download cable. Alternatively, you can program EPCS or EPCQ device using a microprocessor with the SRunner software driver.

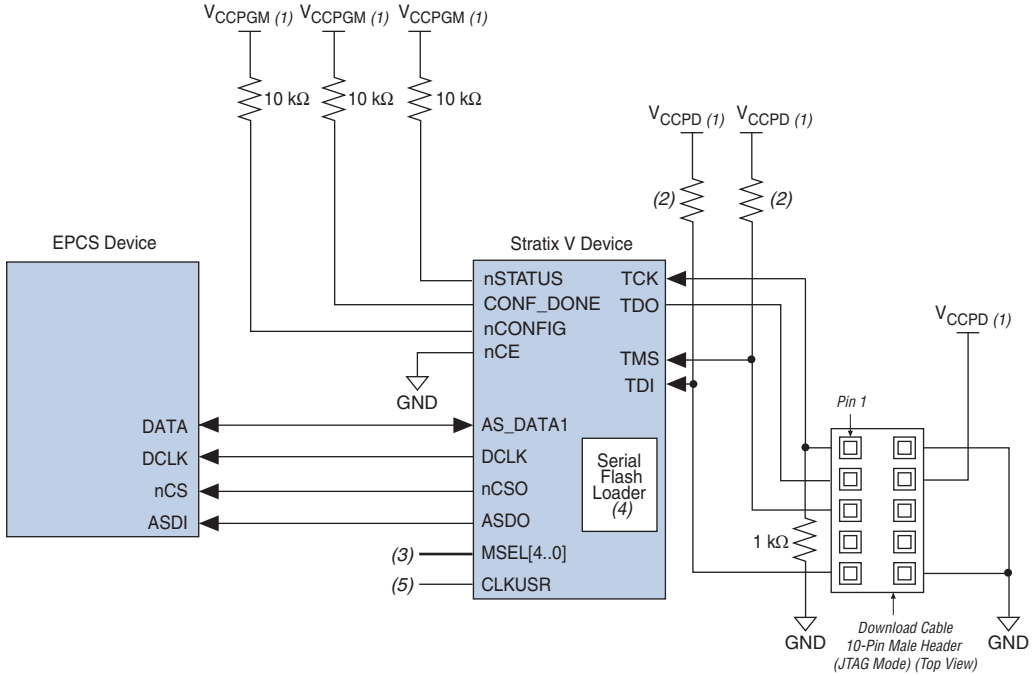
If you are not using Quartus II software or SRunner software for EPCQ 256 programming, put your EPCQ 256 device into four-byte addressing mode before you program and configure your device.

For more information about SRunner software driver, refer to *AN 418: SRunner: An Embedded Solution for Serial Configuration Device Programming*.

In-system programming offers you an option to program the EPCS or EPCQ device either using an AS programming interface or a JTAG interface. Using the AS programming interface, the configuration data is programmed into the EPCS by the Quartus II software or any supported third-party software. Using the JTAG interface, an Altera IP called Serial Flash Loader (SFL) must be downloaded into the Stratix V device to form a bridge between the JTAG interface and the EPCS or EPCQ device. This allows the EPCS or the EPCQ device to be programmed directly using the JTAG interface.

Figure 9–11 shows the connection setup when programming the EPCS device using the JTAG interface.

Figure 9–11. Connection Setup for Programming the EPCS Device Using the JTAG Interface



Notes to Figure 9–11:

(1) Connect the pull-up resistors to V_{CCPGM} and V_{CCPD} at a 3.0-V supply.

(2) Resistor value can vary from 1 kΩ to 10 kΩ. Perform signal integrity analysis to select the resistor value for your setup.

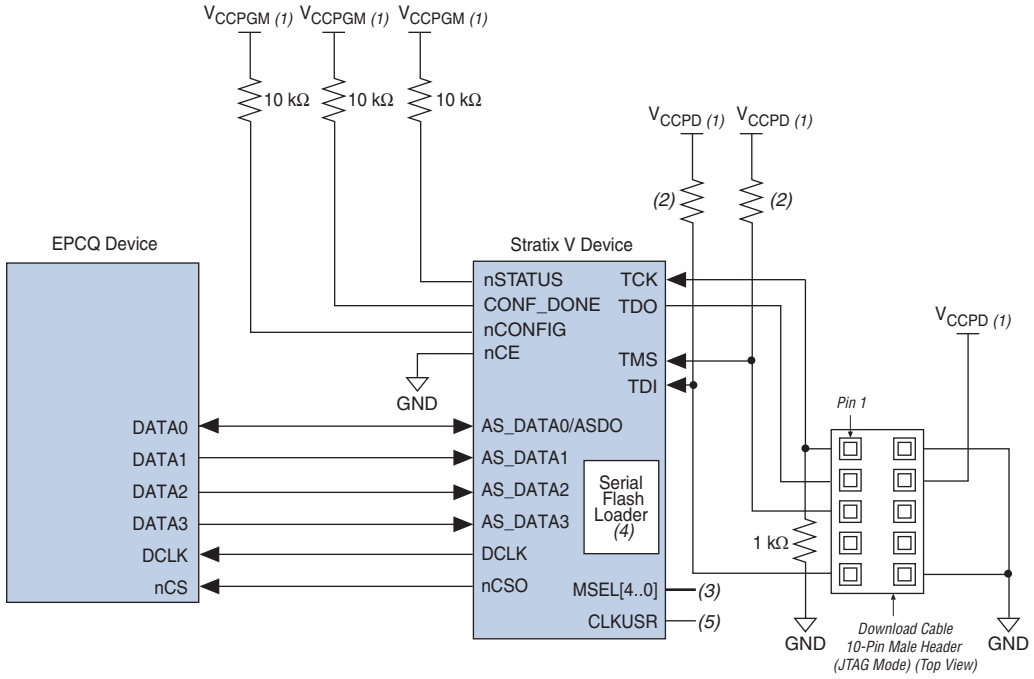
(3) The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect the MSEL, refer to Table 9–4 on page 9–7.

(4) Instantiate SFL in your design to form a bridge between the EPCS device and the Stratix V device. For more information about SFL, refer to *AN 370: Using the Serial Flash Loader with the Quartus II Software*.

(5) You can use the CLKUSR pin to supply the external clock source to drive the DCLK during configuration. The maximum frequency specification is 100 MHz.

Figure 9–12 shows the connection setup when programming the EPCQ device using the JTAG interface.

Figure 9–12. Connection Setup for Programming the EPCQ Device Using the JTAG Interface



Notes to Figure 9–12:

(1) Connect the pull-up resistors to V_{CCPGM} and V_{CCPD} at a 3.0-V supply.

(2) Resistor value can vary from 1 kΩ to 10 kΩ. Perform signal integrity analysis to select the resistor value for your setup.

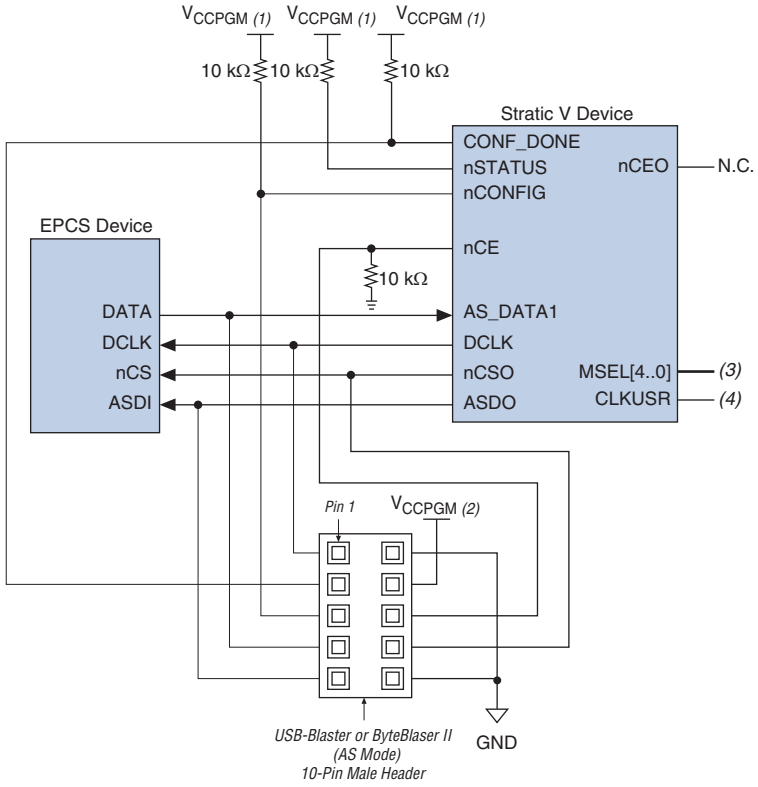
(3) The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect the MSEL, refer to Table 9–4 on page 9–7.

(4) Instantiate SFL in your design to form a bridge between the EPCS device and the Stratix V device. For more information about SFL, refer to *AN 370: Using the Serial Flash Loader with the Quartus II Software*.

(5) You can use the CLKUSR pin to supply the external clock source to drive the DCLK during configuration. The maximum frequency specification is 100 MHz.

Figure 9–13 shows the connection setup when programming the EPCS device using the AS interface.

Figure 9–13. Connection Setup for Programming the EPCS Device Using the AS Interface



Notes to Figure 9–13:

(1) Connect the pull-up resistors to V_{CCPGM} and V_{CCPD} at a 3.0-V supply.

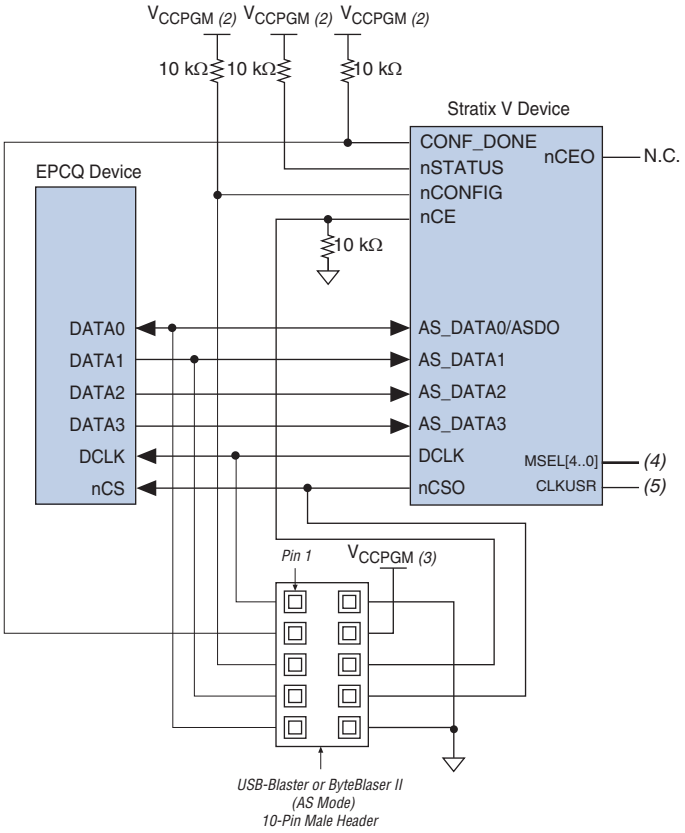
(2) Power up the USB-ByteBlaster, ByteBlaster II, or EthernetBlaster cable's $V_{CC(TRGT)}$ with V_{CCPGM}.

(3) The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect the MSEL, refer to Table 9–4 on page 9–7.

(4) You can use the CLKUSR pin to supply the external clock source to drive the DCLK during configuration. The maximum frequency specification is 100 MHz.

Figure 9–14 shows the connection setup when programming the EPCQ device using the AS interface.

Figure 9–14. Connection Setup for Programming the EPCQ Device Using the AS Interface ***(Note 1)***



Notes to Figure 9–14:

(1) Using the AS header, the programmer transmits the operation commands and the configuration bits to the EPCQ device serially on DATA0.This is equivalent to the programming operation for the EPCS device as shown in Figure 9–14.

(2) Connect the pull-up resistors to V_{CCPGM} and V_{CCPD} at a 3.0-V supply.

(3) Power up the USB-ByteBlaster, ByteBlaster II, or EthernetBlaster cable's $V_{CC(TRGT)}$ with V_{CCPGM}.

(4) The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect the MSEL, refer to Table 9–4 on page 9–7.

(5) You can use the CLKUSR pin to supply the external clock source to drive the DCLK during configuration. The maximum frequency specification is 100 MHz.

During EPCS and EPCQ programming, the download cable disables the device access to the AS interface by driving the nCE pin high. The nCONFIG line is also pulled low to hold the Stratix V device in reset stage. After programming completes, the download cable releases nCE and nCONFIG, allowing the pull-down and pull-up resistors to drive the pin to GND and V_{CCPGM}, respectively.

During EPCQ programming using the download cable, DATA0 carries the programming data, operation command, and address information from the download cable into the EPCQ device. During EPCQ verification using the download cable, DATA1 carries the programming data back to the download cable.

Passive Serial Configuration

You can perform PS configuration of Stratix V devices using an external host such as a MAX II device, microprocessor, or a host PC. Therefore, the design that controls the configuration stages, such as fetching the data from flash memory and sending it to the device, must be stored in the external host.

The Parallel Flash Loader (PFL) feature in MAX II devices provide an efficient method to program CFI flash memory devices through the JTAG interface. PFL also acts as a controller to read configuration data from the flash memory device and configures the Stratix V device. PFL supports both the PS and FPP configuration schemes.

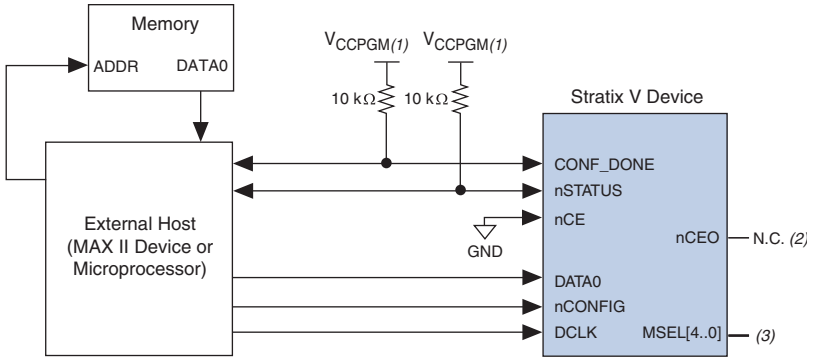
For more information about the PFL, refer to the *Parallel Flash Loader Megafunction User Guide*.

PS Configuration Using a MAX II Device or Microprocessor

The external host (a MAX II device or microprocessor) reads configuration data from storage devices, such as flash memory, and transfers it to Stratix V devices. You can store the configuration data in **.pof**, **.rbf**, **.hex**, or **.ttf** format. If you are using configuration data in **.rbf**, **.hex**, or **.ttf** format, you must send the LSB of each data byte first. For example, if the **.rbf** file contains the byte sequence 02 1B EE 01 FA, the serial data transmitted to the device must be 0100-0000 1101-1000 0111-0111 1000-0000 0101-1111.

Figure 9–15 shows the configuration interface connections between a Stratix V device and a MAX II device for single device configuration.

Figure 9–15. Single Device PS Configuration Using an External Host



Notes to Figure 9–15:

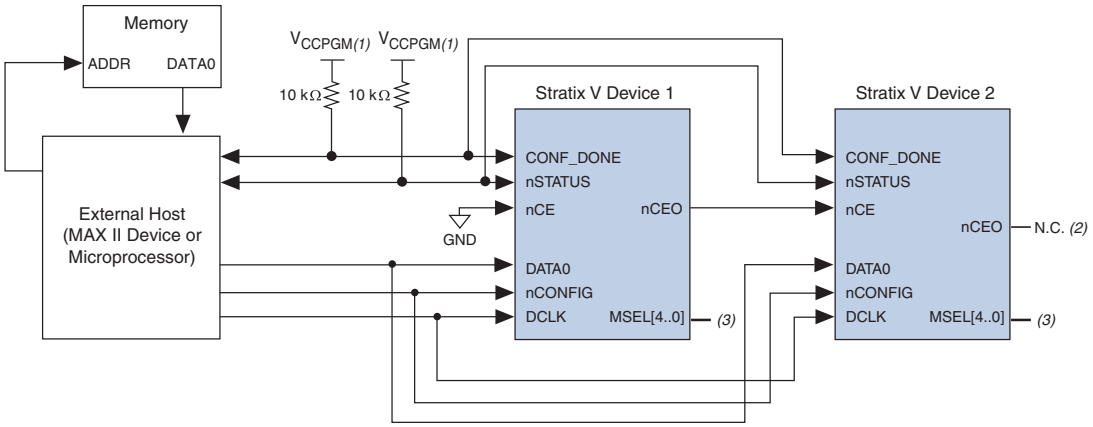
(1) Connect the resistor to a supply that provides an acceptable input signal for the Stratix V device. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends powering up all the configuration system I/Os with V_{CCPGM}.

(2) You can leave the `nCEO` pin unconnected or use it as a user I/O pin when it does not feed another device's `nCE` pin.

(3) The `MSEL` pin settings vary for different configuration voltage standards and POR delays. To connect `MSEL`, refer to Table 9–4 on page 9–7.

Figure 9–16 shows the PS multi-device configuration using an external host when all devices in the chain receive different sets of configuration data (multiple **.sof**s).

Figure 9–16. PS Multi-device Configuration when Both Devices Receive Different Sets of Configuration Data



Notes to Figure 9–16:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Stratix V device. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends powering up all the configuration system I/Os with V_{CCPGM}.

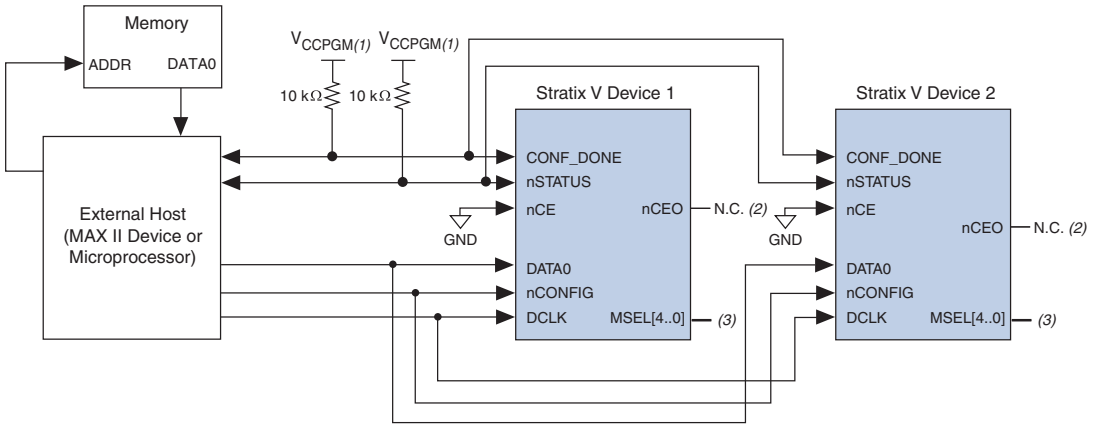
(2) You can leave the nCEO pin unconnected or use it as a user I/O pin when it does not feed another device's nCE pin.

(3) The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect MSEL, refer to Table 9–4 on page 9–7.

In Figure 9–16, after the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration in one clock cycle; therefore, the transfer of data to the second device is transparent to the external host.

Figure 9–17 shows the PS multi-device configuration when all devices receive the same set of configuration data (single **.sof**).

Figure 9–17. PS Multi-device Configuration When Both Devices Receive the Same Set of Configuration Data



Notes to Figure 9–17:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Stratix V device. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends powering up all the configuration system I/Os with V_{CCPGM}.

(2) You can leave the nCEO pin unconnected or use it as a user I/O pin.

(3) The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect MSEL, refer to Table 9–4 on page 9–7.

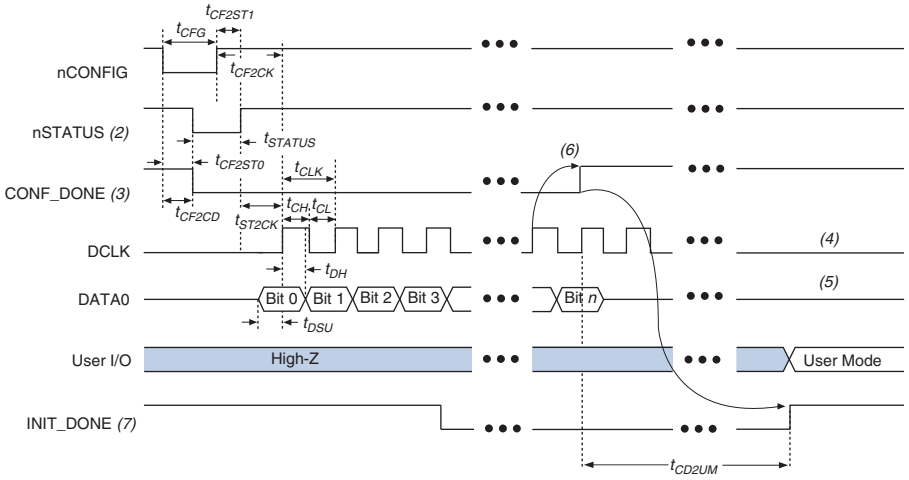
In Figure 9–17, because both nCE pins are tied to GND, both devices in the chain begin and complete the configuration and enter user mode at the same time.

To configure the PS multi-device with a single **.sof**, as shown in Figure 9–17, all Stratix V devices in the chain must be in the same package and density.

PS Configuration Timing

Figure 9–18 shows the timing waveform for a PS configuration when using a MAX II device or microprocessor as an external host.

Figure 9–18. PS Configuration Timing Waveform ***(Note 1)***



Notes to Figure 9–18:

(1) The beginning of this waveform shows the device in user mode. In user mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.

(2) After power-up, the Stratix V device holds nSTATUS low for the time of the POR delay.

(3) After power-up, before and during configuration, CONF_DONE is low.

(4) Do not leave DCLK floating after configuration. You can drive it high or low, whichever is more convenient.

(5) DATA0 is available as a user I/O pin after configuration. The state of this pin depends on the dual-purpose pin settings in the **Device and Pins Option**.

(6) To ensure a successful configuration, send the entire configuration data to the Stratix V device. CONF_DONE is released high after the Stratix V device receives all the configuration data successfully. After CONF_DONE goes high, send two additional falling edges on DCLK to begin initialization and enter user mode.

(7) After the option bit to enable the INIT_DONE pin is configured into the device, the INIT_DONE goes low.

Table 9–12 lists the PS configuration timing parameters for Stratix V devices.

Table 9–12. PS Timing Parameters for Stratix V Devices ***(Note 1)*** **(Part 1 of 2)**

Symbol	Parameter	Minimum	Maximum	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low	—	600	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low	—	600	ns
t_{CFG}	nCONFIG low pulse width	2	—	µs
t_{STATUS}	nSTATUS low pulse width	268	1,506 *(2)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high	—	1,506 *(3)*	µs

Table 9–12. PS Timing Parameters for Stratix V Devices *(Note 1)* (Part 2 of 2)

Symbol	Parameter	Minimum	Maximum	Units
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	1,506	—	µs
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2	—	µs
t_{DSU}	DATA[] setup time before rising edge on DCLK	5.5	—	ns
t_{DH}	DATA[] hold time after rising edge on DCLK	0	—	ns
t_{CH}	DCLK high time	$0.45 \times 1/f_{MAX}$	—	s
t_{CL}	DCLK low time	$0.45 \times 1/f_{MAX}$	—	s
t_{CLK}	DCLK period	$1/f_{MAX}$	—	s
f_{MAX}	DCLK frequency	—	125	MHz
t_R	Input rise time	—	40	ns
t_F	Input fall time	—	40	ns
t_{CD2UM}	CONF_DONE high to user mode *(4)*	175	437	µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period	—	—
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	t_{CD2CU} + (17,408 × CLKUSR period) *(5)*	—	—

Notes to Table 9–12:

(1) This information is preliminary.
(2) This value is applicable if you do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.
(3) This value is applicable if you do not delay configuration by externally holding the nSTATUS low.
(4) The minimum and maximum numbers apply only if you choose the internal oscillator as the clock source for initializing the device.
(5) To enable the CLKUSR pin as the initialization clock source and to obtain the maximum frequency specification on these pins, refer to "Initialization" on page 9–5.

Two DCLK falling edges are required after CONF_DONE goes high to begin the initialization of the device for both uncompressed and compressed configuration data in the PS configuration scheme.

PS Configuration Using a Download Cable

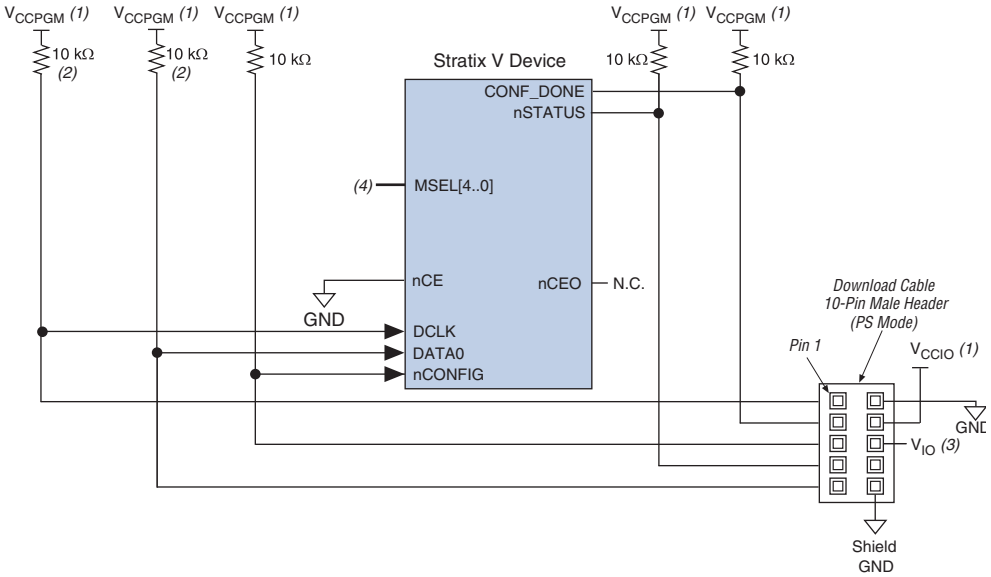
In this section, the generic term "download cable" includes the Altera USB-Blaster universal serial bus (USB) port download cable, ByteBlaster II parallel port download cable, and EthernetBlaster download cable.

In a PS configuration with a download cable, a PC acts as a host to transfer data from a storage device to the Stratix V device using the download cable. During configuration, the programming hardware or download cable places the configuration data one bit at a time on the device's DATA0 pin. The configuration data is clocked into the target device until CONF_DONE goes high.

If you turn on the CLKUSR option during PS configuration using a download cable and the Quartus II programmer, you do not have to provide a clock on the CLKUSR pin to initialize your device.

Figure 9–19 shows a PS configuration for Stratix V devices using an Altera download cable.

Figure 9–19. PS Configuration Using an Altera Download Cable



Notes to Figure 9–19:

(1) Connect the pull-up resistor to the same supply voltage (V_{CCIO}) as the USB-Blaster, ByteBlaster II, or EthernetBlaster cable.

(2) You only need the pull-up resistors on DATA0 and DCLK if the download cable is the only configuration scheme used on your board. This ensures that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a MAX II device or microprocessor, you do not need the pull-up resistors on DATA0 and DCLK.

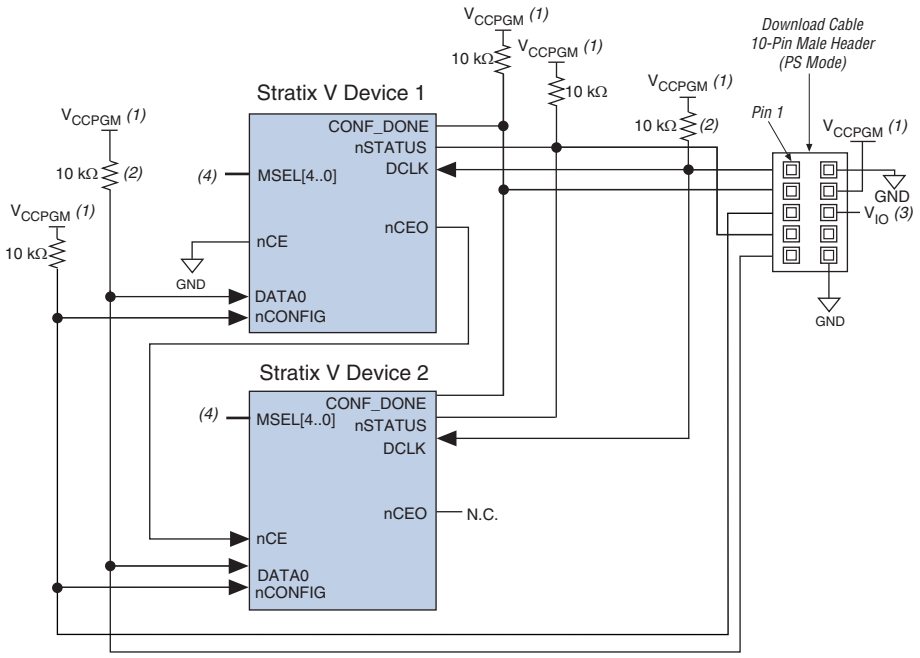
(3) In the USB-Blaster and ByteBlaster II cables, this pin is connected to nCE when you use for AS programming; otherwise, it is a no connect.

(4) The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect MSEL[4..0], refer to Table 9–4 on page 9–7.

Multi-Device PS Configuration Using Download Cable

You can use a download cable to configure multiple Stratix V devices as shown in Figure 9–20.

Figure 9–20. Multi-Device PS Configuration Using an Altera Download Cable



Notes to Figure 9–20:

(1) Connect the pull-up resistor to the same supply voltage (V_{CCIO}) as the USB-Blaster, ByteBlaster II, or EthernetBlaster cable.

(2) You only need the pull-up resistors on DATA0 and DCLK if the download cable is the only configuration scheme used on your board. This ensures that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, you do not need the pull-up resistors on DATA0 and DCLK.

(3) In the USB-Blaster and ByteBlaster II cables, this pin is connected to nCE when you use it for AS programming; otherwise, it is a no connect.

(4) The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect MSEL[4..0], refer to Table 9–4 on page 9–7.

In Figure 9–20, after the first device completes configuration, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE pins are connected to every device in the chain. As all the CONF_DONE and nSTATUS pins are tied together, all devices initialize and enter user mode at the same time. If any device detects an error, configuration stops for the entire chain and you must reconfigure all the devices. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

JTAG Configuration

You can use the same JTAG interface specifically developed for boundary-scan test (BST) to shift the configuration data into the device. The Quartus II software automatically generates a **.sof** that you can use for JTAG configuration with a download cable in the Quartus II software programmer.

For more information about JTAG BST and the commands available using Stratix V devices, refer to the following documents:

- *JTAG Boundary-Scan Testing in Stratix V Devices* chapter
- Programming Support for Jam STAPL Language

Stratix V devices are designed such that JTAG instructions have precedence over any device configuration modes. JTAG configuration can take place without waiting for other configuration modes to complete. For example, if you attempt JTAG configuration of Stratix V devices during a PS configuration, the PS configuration is terminated and the JTAG configuration begins. All user I/O pins are tri-stated during the JTAG configuration.

You cannot use the Stratix V decompression or design security features if you are configuring your Stratix V device using JTAG-based configuration.

For more information about TDI, TDO, TMS, and TCK, refer to "Device Configuration Pins" on page 9–38.

For instructions to connect a JTAG chain with multiple voltages across the devices in the chain, refer to the *JTAG Boundary Scan Testing in Stratix V Devices* chapter.

To configure a single device in a JTAG chain, the programming software places all the other devices in bypass mode. In bypass mode, devices pass programming data from the TDI pin to the TDO pin through a single bypass register without being affected internally. This scheme enables the programming software to program or verify the target device. Configuration data driven into the device appears on the TDO pin one clock cycle later. The Quartus II software verifies successful JTAG configuration after completion by checking the state of CONF_DONE through the JTAG port.

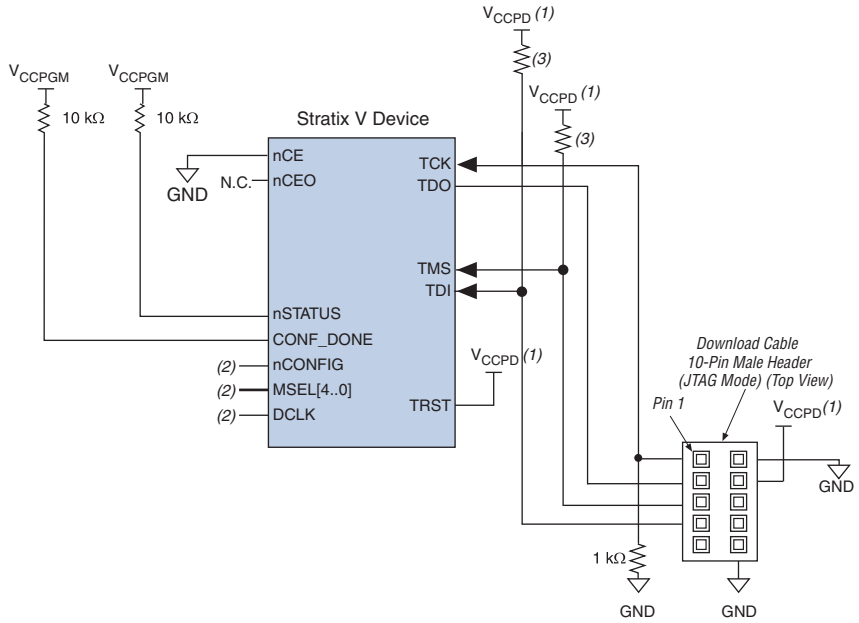
If CONF_DONE is not high, the Quartus II software indicates that configuration has failed. If CONF_DONE is high, the software indicates that configuration was successful. After the configuration data is transmitted serially using the JTAG TDI port, the TCK port is clocked an additional 1,222 cycles to perform device initialization.

The chip-wide reset (DEV_CLRn) and chip-wide output enable (DEV_OE) pins on Stratix V devices do not affect JTAG boundary-scan or programming operations.

You can generate a JAM File (**.jam**) or Jam-byte Code (**.jbc**) to be used with other third party programmer tools. Alternatively, you can use JRunner with **.rbf** to program your device.

Figure 9–21 shows the JTAG configuration of a single Stratix V device.

Figure 9–21. JTAG Configuration of a Single Device Using a Download Cable



Notes to Figure 9–21:

(1) Connect the pull-up resistor V_{CCPD}. For more information, refer to V_{CCPD} requirement in "V_{CCPD} Pin" on page 9–3.

(2) If you only use JTAG configuration, connect `nCONFIG` to V_{CCPGM} and `MSEL[4..0]` to GND. Pull `DCLK` either high or low, whichever is convenient on your board. If you are using JTAG in conjunction with another configuration scheme, connect `MSEL[4..0]`, `nCONFIG` and `DCLK` based on the selected configuration scheme.

(3) The resistor value can vary from 1 kΩ to 10 kΩ. Perform signal integrity analysis to select the resistor value for your setup.

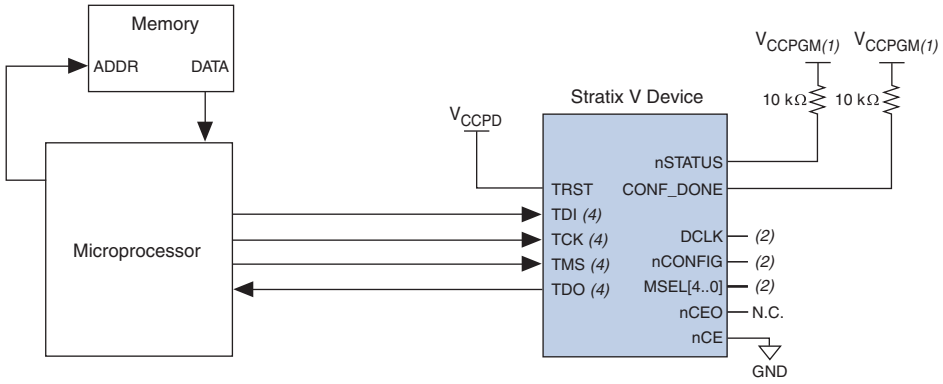
(4) You must connect `nCE` to GND or drive it low for successful JTAG configuration.

Alternatively, you can use a microprocessor to program the device through the JTAG interface. You can use JRunner as your software driver.

For more information about JRunner, refer to *AN 414: The JRunner Software Driver: An Embedded Solution for PLD JTAG Configuration*.

Figure 9–22 shows a JTAG configuration of a Stratix V device using a microprocessor.

Figure 9–22. JTAG Configuration of a Single Device Using a Microprocessor



Notes to Figure 9–22:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for all Stratix V devices in the chain. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device.

(2) If you only use the JTAG configuration, connect nCONFIG to V_{CCPGM} and MSEL[4..0] to GND. Pull DCLK either high or low, whichever is convenient on your board. If you are using JTAG in conjunction with another configuration scheme, set the MSEL[4..0] and tie nCONFIG and DCLK based on the selected configuration scheme.

(3) Connect nCE to GND or drive it low for successful JTAG configuration.

(4) The microprocessor must use the same I/O standard as V_{CCPD} to drive the JTAG pins.

CONFIG_IO Instruction

The CONFIG_IO instruction allows you to configure I/O buffers using the JTAG port and when issued, interrupts configuration. This instruction allows you to perform board-level testing prior to configuring the Stratix V device or waiting for a configuration device to complete configuration. After configuration is interrupted and JTAG testing is complete, you must reconfigure the part using the JTAG interface or if you support FPP, PS, or AS on your board, reconfigure the device by externally pulsing nCONFIG low. Alternatively, you can pulse nCONFIG low through the same JTAG interface using the PULSE_NCONFIG JTAG instruction.

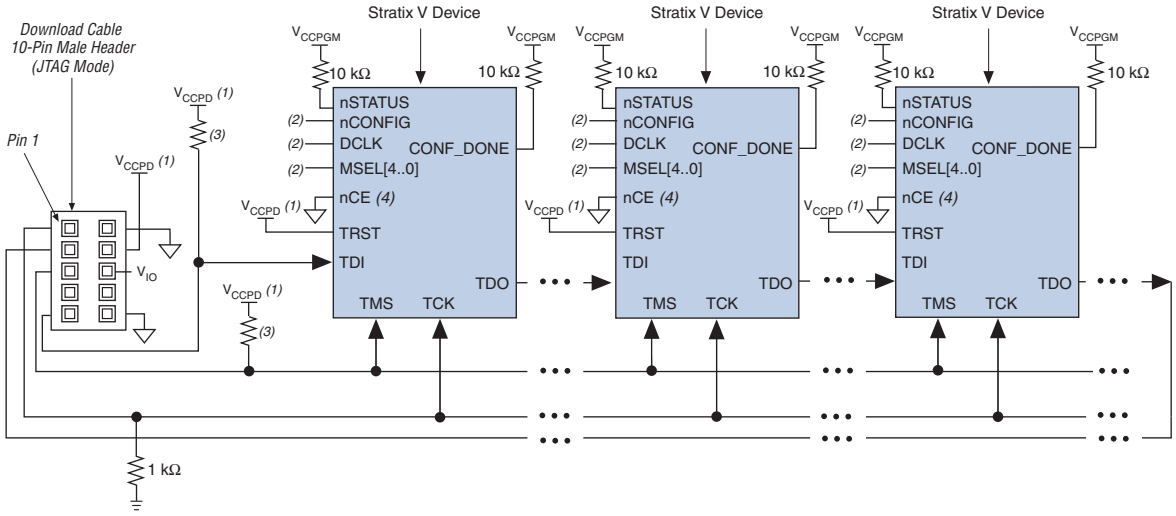
All JTAG instructions (except BYPASS, IDCODE, and SAMPLE) can be issued by first interrupting the configuration and reprogramming the I/O pins using the CONFIG_IO instruction.

Multi-Device JTAG Configuration

When programming a JTAG device chain, one JTAG-compatible header is connected to several devices. The number of devices in the JTAG chain is limited only by the drive capability of the download cable. When four or more devices are connected in a JTAG chain, Altera recommends buffering the TCK, TDI, and TMS pins with an on-board buffer. You can place other Altera devices that have JTAG support in the same JTAG chain for device programming.

JTAG-chain device programming is ideal when the system contains multiple devices or when testing your system using JTAG BST circuitry. Figure 9–23 shows a multi-device JTAG configuration.

Figure 9–23. JTAG Configuration of Multiple Devices Using a Download Cable



Notes to Figure 9–23:

(1) Connect the pull-up resistor V_{CCPD}. For more information, refer to V_{CCPD} requirement in "V_{CCPD} Pin" on page 9–3.

(2) If you only use JTAG configuration, connect `nCONFIG` to V_{CCPGM} and `MSEL[4..0]` to GND. Pull `DCLK` either high or low, whichever is convenient on your board. If you are using JTAG in conjunction with another configuration scheme, connect the `MSEL[4..0]`, `nCONFIG`, and `DCLK` based on the selected configuration scheme.

(3) The resistor value can vary from 1kΩ to 10 kΩ. Perform signal integrity analysis to select the resistor value for your setup.

(4) You must connect `nCE` to GND or drive it low for successful JTAG configuration.

If you want to use the JTAG multi-device configuration in conjunction with other schemes, such as a FPP, PS, or AS, tie `CONF_DONE`, `nSTATUS`, and `nCONFIG` together as recommended in the FPP, PS, or AS multi-device configuration schemes. Ensure that the JTAG chain is the same order as the multi-device FPP, PS, or AS configuration chain.

If you only use JTAG configuration, Altera recommends connecting the circuitry as shown in Figure 9–22, where each of the `CONF_DONE` and `nSTATUS` signals are isolated to enable each device to enter user mode individually.

For more information about combining the JTAG configuration with other configuration schemes, refer to the *Combining Different Configuration Schemes* chapter in volume 2 of the *Configuration Handbook*.

For more information about JTAG and Jam STAPL in embedded environments, refer to *AN 425: Using Command-Line Jam STAPL Solution for Device Programming*. To download the Jam player, visit the Altera website.

For more information about how to use the USB-Blaster, ByteBlaster II, or EthernetBlaster cables, refer to the following user guides:

- *USB-Blaster Download Cable User Guide*
- *ByteBlaster II Download Cable User Guide*
- *Ethernet Blaster Communications Cable User Guide*

Device Configuration Pins

Table 9–13 and Table 9–14 list the connections and functionality of all the configuration-related pins on the Stratix V devices. Table 9–13 lists the Stratix V configuration pins and their power supply.

Table 9–13. Configuration Pin Summary for Stratix V Devices (Part 1 of 2)

Description	Input/Output	User Mode	Powered By	Configuration Scheme
TDI	Input	—	V_{CCPD}	JTAG
TMS	Input	—	V_{CCPD}	JTAG
TCK	Input	—	V_{CCPD}	JTAG
TRST	Input	—	V_{CCPD}	JTAG
TDO	Output	—	V_{CCPD}	JTAG
CLKUSR	Input	I/O *(1)*	V_{CCPGM}/V_{CCIO} *(4)*	All schemes
CRC_ERROR	Output	I/O *(1)*	Pull-up	Optional, all schemes
CONF_DONE	Bidirectional	—	V_{CCPGM}/Pull-up	All schemes
DATA0	Bidirectional	I/O *(2)*	V_{CCPGM}/V_{CCIO} *(4)*	FPP, PS
DATA[31..1]	Bidirectional	I/O *(2)*	V_{CCPGM}/V_{CCIO} *(4)*	FPP
DCLK	Input	—	V_{CCPGM}	FPP, PS
	Output	—	V_{CCPGM}	AS
DEV_OE	Input	I/O *(1)*	V_{CCPGM}/V_{CCIO} *(4)*	Optional, all schemes
DEV_CLRn	Input	I/O *(1)*	V_{CCPGM}/V_{CCIO} *(4)*	Optional, all schemes
INIT_DONE	Output	I/O *(1)*	Pull-up	Optional, all schemes
MSEL[4..0]	Input	—	V_{CCPGM}	All schemes
nSTATUS	Bidirectional	—	V_{CCPGM}	All schemes
nCE	Input	—	V_{CCPGM}	All schemes
nCEO	Output	I/O *(3)*	Pull-up	All schemes
nCONFIG	Input	—	V_{CCPGM}	All schemes
nCSO	Output	—	V_{CCPGM}	AS
nIO_PULLUP	Input	—	V_{CC} *(5)*	All schemes
AS_DATA0/ASDO	Bidirectional	—	V_{CCPGM}	AS

Table 9–13. Configuration Pin Summary for Stratix V Devices (Part 2 of 2)

Description	Input/Output	User Mode	Powered By	Configuration Scheme
AS_DATA[3..1]	Bidirectional	—	V_{CCPGM}	AS

Notes to Table 9–13:

(1) This is a dual-purpose pin. This pin is available as an I/O if the associated option that enables this pin is turned off from the **Configuration** panel in the **Device and Pins Option** settings. For example, the DEV_OE is available as a user I/O if the **Enable device-wide output enable** option is turned off.

(2) This is a dual-purpose pin. The state of this pin in user mode depends on the **Dual-purpose Pins** settings in the **Device and Pins Option**.

(3) This pin is available as an I/O if this pin is not feeding the next device's nCE in a multi-device configuration. To use this pin to feed the next device's nCE in a multi-device chain, turn on **Enable INIT_DONE output** option under **Device and Pins Option**, **General** panel in the Quartus II Software.

(4) This pin is powered up by V_{CCPGM} during configuration. It is powered up by V_{CCIO} of the bank in which the pin resides if it is used as a regular I/O in user mode.

(5) Although nIO_PULLUP is powered up by V_{CC}, Altera recommends connecting this pin to V_{CCPGM} or GND directly without using a pull-up or pull-down resistor.

Table 9–14 lists the configuration pin descriptions.

Table 9–14. Configuration Pins Description (Part 1 of 3)

Pin Name	Description
TDI *(1)*	Dedicated test data input. Serial input pin for instructions as well as test and programming data. Data is shifted on the rising edge of TCK. This pin has an internal 25-kΩ pull-up that is always active.
TMS *(1)*	Dedicated test mode select. Input pin that provides the control signal to determine the transitions of the TAP controller state machine. TMS is evaluated on the rising edge of TCK. Therefore, you must set up TMS before the rising edge of TCK. Transitions in the state machine occur on the falling edge of TCK after the signal is applied to TMS. This pin has an internal 25-kΩ pull-up that is always active.
TCK *(1)*	Dedicated test clock input. Clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. It is expected that the clock input waveform have a nominal 50% duty cycle. This pin has an internal 25-kΩ pull-down that is always active.
TRST *(1)*	Dedicated test reset input. Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to the IEEE Std. 1149.1 standard. Connecting this pin low disables the JTAG circuitry. This pin has an internal 25-kΩ pull-up that is always active.
TDO *(1)*	Dedicated test data output. Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. This pin is tri-stated if the data is not being shifted out of the device.
CLKUSR	Optional user-supplied clock input. It synchronizes the initialization of one or more devices. Enable this pin by turning on the **Enable user-supplied start-up clock (CLKUSR)** option under **Device and Pins Option**, **Configuration** panel in the Quartus II software.

Table 9–14. Configuration Pins Description (Part 2 of 3)

Pin Name	Description
CRC_ERROR	Optional output pin. Signals that the device has detected a cyclical redundancy check (CRC) error during user mode operation. this pin is an open-drain output pin by default and requires a 10 kΩ pull-up resistor. To use this pin as regular output, turn-off the **Enable Open-drain on CRC_ERROR pin** in **Device and Pins Option**, **Error Detection CRC** panel in the Quartus II software. The target device drives this pin low if there is no CRC error in user mode operation. As an open-drain output, if a CRC error occurs, the device releases the pin which is then pulled high by the external pull-up resistor. Enable this pin by turning on **Enable CRC error detection on CRC_ERROR pin** option in the Quartus II software. For more information about the CRC_ERROR pin, refer to *SEU Mitigation in Stratix V Devices* chapter.
CONF_DONE	Dedicated open-drain bidirectional pin. The target device drives the CONF_DONE pin low before and during configuration. After all the configuration data is received without error and the initialization cycle starts, the target device releases the CONF_DONE pin, which is then pulled high by the external pull-up resistor. The target device then reads the CONF_DONE pin status to ensure that the CONF_DONE is at logic high. After it is sensed high, the target device initializes and enters user mode. Driving CONF_DONE low after initialization completes does not affect the configured device.
DATA0 *(3)*	Dual-purpose data input pin. The data received on DATA0 is synchronized to DCLK. After configuration completes, this pin is available as a user I/O pin.
DATA[31..1] *(3)*	Dual-purpose data input pins. If you are using FPP ×16 or FPP ×32, only a subset of these pins are required for configuration. The pins that are not required for configuration, you can use them as regular I/Os. During configuration, byte-wide, or word-wide data is received on these pins. The data received on DATA[31..1] are synchronized to the DCLK.
DCLK	Dedicated bidirectional clock pin. In the PS and FPP configurations, DCLK is the clock input used to clock data from an external source into the target device. Data is latched into the device on the rising edge of DCLK. After configuration completes, drive DCLK high or low, whichever is more convenient. In the AS mode, DCLK is an output clock to clock the EPCS or EPCQ devices. Data is latched into the device on the falling edge of the DCLK. After AS configuration completes, this pin is tri-stated with a weak pull-up resistor. Toggling this pin after configuration does not affect the configured device.
DEV_OE *(2)*	Optional input pin that allows you to override all tri-states on the device. When this pin is driven low, all the I/O pins are tri-stated. When this pin is driven high, all the I/O pins behave as programmed. Enable this pin by turning on the **Enable device-wide output enable (DEV_OE)** option in the Quartus II software.
DEV_CLRn *(2)*	Optional input pin that allows you to override all clears on all the device registers. When this pin is driven low, all the registers are cleared. When this pin is driven high, all the registers behave as programmed. This pin is enabled by turning on the **Enable device-wide reset (DEV_CLRn)** option in the Quartus II software.
INIT_DONE *(2)*	Optional output pin. Signals when the device has initialized and is in user mode. During the reset stage, after the device exits POR, and during the beginning of the configuration, the INIT_DONE pin is tri-stated and pulled high due to an external pull-up resistor. After the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization completes, the INIT_DONE pin is released and pulled high and the device enters user mode. Thus, the monitoring circuitry must be able to detect a low-to-high transition. Enable this pin by turning on the **Enable INIT_DONE output** option in the Quartus II software.

Table 9–14. Configuration Pins Description (Part 3 of 3)

Pin Name	Description
MSEL[4..0]	Dedicated input pins. Five-bit configuration input that sets the Stratix V device configuration scheme. For the appropriate connections, refer to Table 9–4 on page 9–7. The MSEL[4..0] pins have internal 5-kΩ pull-down resistors that are always active.
nSTATUS	Dedicated open-drain bidirectional pin. The device drives nSTATUS low immediately after power-up and releases it after the device exits POR. During user mode and regular configuration, this pin is pulled high by an external 10-kΩ resistor. During configuration, the device drives this pin low to indicate an error during configuration. If an external source drives the nSTATUS pin low during configuration or initialization, the target device enters an error state. This mechanism is used during multi-device configuration setup. If one of the devices in the chain has an error and pulls its nSTATUS pin low, it resets the entire chain. Driving nSTATUS low after configuration and initialization completes does not affect the configured device.
nCE	Dedicated active-low chip enable input pin. Driving this pin low allows configuration. Drives the nCE pin low during configuration, initialization, and user mode for all single-device configurations. For a multi-device configuration, connect the nCE pin to GND or to the nCEO of the previous device in the chain based on the recommendation in the respective configuration setup diagram.
nCEO *(3)*	Dual-purpose open-drain output pin. This pin drives low when device configuration completes. To use this pin to feed the next device's nCE pin in a multi-device chain, turn on **Enable INIT_DONE output** option under **Device and Pins Option**, **General** panel in the Quartus II Software. In a single-device configuration, this pin can be used as a regular I/O. In multi-device configuration, if this pin is not feeding the nCE of the next device, you can use it as a regular I/O.
nCONFIG	Dedicated input pin. A low pulse on this pin during configuration and user mode causes the device to enter a reset state and tri-states all the I/O pins. A low-to-high logic starts a reconfiguration. During JTAG programming, the nCONFIG status is ignored.
nCSO	Dedicated output pin. Drives the control signal from the Stratix V device to the EPCS and EPCQ devices in AS mode. After AS configuration completes, these pins are tri-stated with a weak pull-up resistor.
nIO_PULLUP	Dedicated input pin. This input pin enables or disables the internal pull-up resistors on the user I/O pins and dual purpose I/O pins (DATA[31..0], CLKUSR, INIT_DONE, DEV_OE, and DEV_CLRN). A logic high turns off the weak internal pull-up resistors, while a logic low turns them on. This pin has an internal 5-kΩ pull-down resistor that is always active.
AS_DATA0/ASDO	Dedicated bidirectional data pin. In AS ×1 and AS ×4 configurations, ASDO is used to send the operation command and addresses to the EPCS or EPCQ devices. During an AS ×4 configuration, the data is received on an AS_DATA0 and is synchronized to DCLK. After AS configuration completes, this pin is tri-stated with a weak pull-up resistor.
AS_DATA[3..1]	Dedicated bidirectional data pins. During an AS configuration, the data is received on these pins and is synchronized to DCLK. After AS configuration completes, these pins are tri-stated with a weak pull-up resistor.

Notes to Table 9–14:

(1) If the JTAG interface is not required on the board, you can disable the JTAG circuitry by connecting this pin to logic high. For instructions to connect a JTAG chain with multiple voltages across the devices in the chain, refer to the *JTAG Boundary Scan Testing* chapter.

(2) This is a dual-purpose pin. This pin is available as an I/O if the associated option that enables this pin is turned off from the **Configuration** panel in the **Device and Pins Option** settings. For example, the DEV_OE is available as a user I/O if the **Enable device-wide output enable** option is turned off.

(3) This is a dual-purpose pin. The state of this pin in the user mode depends on **Dual-purpose Pins** settings in the **Device and Pins Option** settings.

Configuration Data Decompression

Stratix V devices support configuration data decompression, which saves configuration memory space and may shorten the configuration time. This feature allows you to store compressed configuration data in the configuration or other memory devices and transmit this compressed data to the Stratix V devices. During configuration, the Stratix V device decompresses the data in real time and programs its SRAM cells. The data decompression is done on-the-fly during configuration and does not require an additional processing time.

Preliminary data indicates that compression typically reduces the configuration data size by 30 to 55% based on the designs used. This reduces the storage requirement capacity for the flash memory. The decompression feature is supported in all configuration schemes except JTAG.

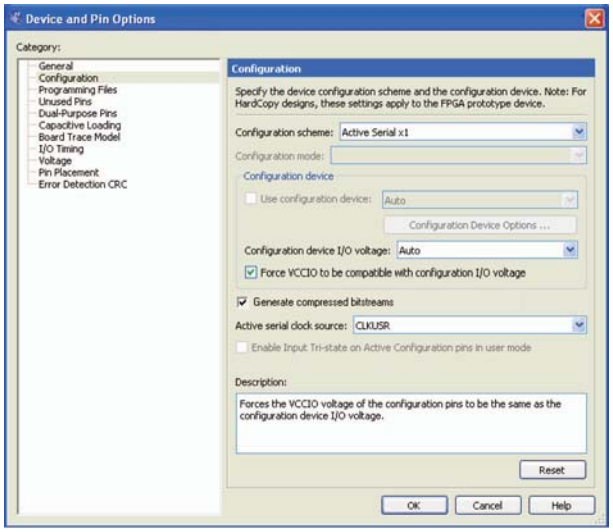
In FPP, enabling the decompression feature requires a different DCLK-to-DATA[] ratio. For more information, refer to "Fast Passive Parallel Configuration" on page 9–9.

There are two ways to enable compression for Stratix V data—before design compilation (in the Compiler Settings menu) and after design compilation (in the Convert Programming Files window).

To enable compression in the project's Compiler Settings menu, follow these steps:

1. On the Assignments menu, click **Device** to bring up the **Settings** dialog box.
2. After selecting your Stratix V device, open the **Device and Pin Options** dialog box.
3. In the **Configuration settings** panel, turn on the **Generate compressed bitstreams** option (refer to Figure 9–24).

Figure 9–24. Enabling Compression Bitstreams in Compiler Settings for Stratix V



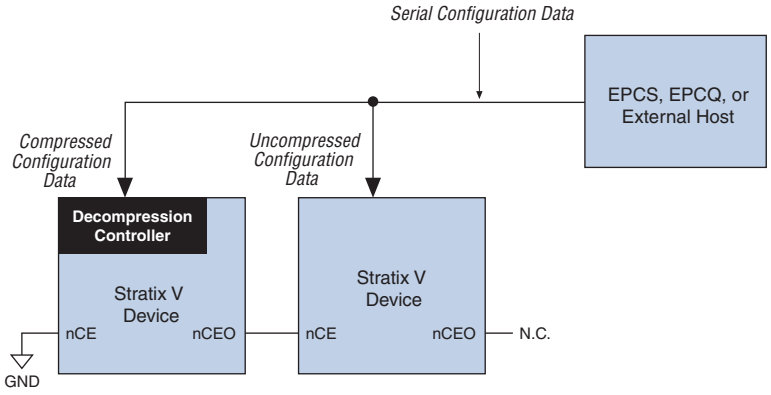
You can also enable compression when creating programming files from the **Convert Programming Files** window. To do this, follow these steps:

1. On the File menu, click **Convert Programming Files**.
2. Select the programming file type (**.pof**, **.sram**, **.hex**, **.hexout**, **.rbf**, or **.ttf**).
3. For POF output files, select a configuration device.
4. In the **Input files to convert** box, select **SOF Data**.
5. Select **Add File** and add a Stratix V device **.sof**.
6. Select the name of the file you added to the **SOF Data** area and click **Properties**.
7. Check the **Compression** check box.

If you are using a serial configuration scheme, AS ×1 or PS, for multi-device configuration, you can selectively enable the compression feature for each device in the chain. Figure 9–25 shows a chain of two Stratix V devices. The first Stratix V device has compression enabled and therefore receives compressed data from the external host. The second Stratix V device has the compression feature disabled and receives uncompressed data.

For FPP configuration schemes, a combination of compressed and uncompressed configuration in the same multi-device chain is not allowed due to the difference of the DCLK-to-DATA[] ratio.

Figure 9–25. Compressed and Uncompressed Serial Configuration Data in the Same Configuration File ***(Note 1)***



Note to Figure 9–25:

(1) The configuration for this setup can be generated from the Convert Programming Files menu in the Quartus II software.

Remote System Upgrades

This section describes the functionality and implementation of the dedicated remote system upgrade circuitry. It also defines several concepts related to remote system upgrades, including factory configuration, application configuration, remote update mode, and user watchdog timer. Additionally, this section provides design guidelines for implementing remote system upgrades with the supported configuration schemes.

System designers sometimes face challenges such as shortened design cycles, evolving standards, and system deployments in remote locations. Stratix V devices help overcome these challenges with their inherent reprogrammability and dedicated circuitry to perform remote system upgrades. Remote system upgrades help deliver feature enhancements and bug fixes without costly recalls, reduce time-to-market, extend product life, and help to avoid system downtime.

Stratix V devices feature dedicated remote system upgrade circuitry. A soft logic (either the Nios® II embedded processor or user logic) implemented in a Stratix V device can download a new configuration image from a remote location, store it in configuration memory, and direct the dedicated remote system upgrade circuitry to start a reconfiguration cycle. The dedicated circuitry performs error detection during and after the configuration process, recovers from any error condition by reverting back to a safe configuration image, and provides error status information.

Remote system upgrades are supported in AS configuration schemes with EPCS and EPCQ devices. You can also implement remote system upgrades in conjunction with advanced Stratix V features such as real-time decompression of configuration data and design security using the advanced design security standard (AES) for secure and efficient field upgrades. The largest EPCS and EPCQ device currently supports 128 Mbits and 256 Mbits configuration data respectively.

Remote system upgrades are supported only in single-device configurations.

The remote system upgrade process in Stratix V devices involves the following steps:

1. A Nios II processor (or user logic) implemented in the Stratix V device logic array receives new configuration data from a remote location. The connection to the remote source uses a communication protocol such as TCP/IP, PCI, user datagram protocol (UDP), UART, or a proprietary interface.
2. The Nios II processor (or user logic) stores this new configuration data in non-volatile configuration memory.
3. The Nios II processor (or user logic) starts a reconfiguration cycle with the new or updated configuration data.
4. The dedicated remote system upgrade circuitry detects and recovers from any errors that might occur during or after the reconfiguration cycle and provides error status information to the user design.

Figure 9–26 shows these remote system upgrade steps.

Figure 9–26. Functional Diagram of the Stratix V Remote System Upgrade Process

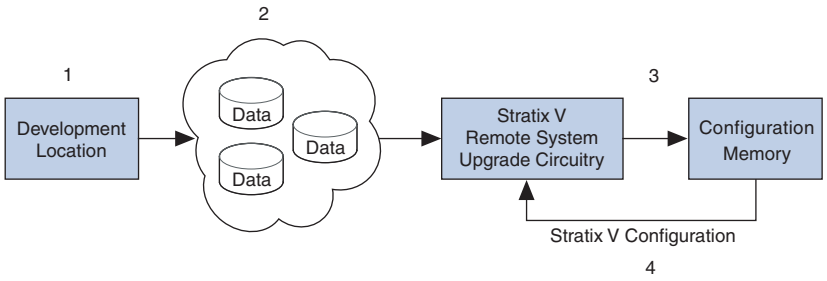
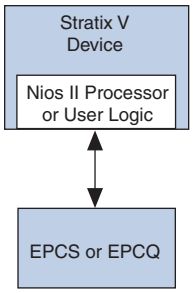


Figure 9–27 shows a block diagram for implementing a remote system upgrade with the Stratix V AS configuration scheme.

Figure 9–27. Remote System Upgrade Block Diagram for Stratix V AS Configuration Scheme *(1)*



Note to Figure 9–27:

(1) You must set the mode select pins (`MSEL[4..0]`) to **AS mode** to use remote system upgrade in your system. The `MSEL` pin settings vary for different POR delays. To connect `MSEL[4..0]`, refer to Table 9–4 on page 9–7.

Configuration Image Types

When performing a remote system upgrade, Stratix V device configuration data are classified as factory configuration images or application configuration images. An image, also referred to as a configuration image, is a design loaded into the Stratix V device that performs certain user-defined functions.

The factory image is a user-defined fall-back, or safe configuration, and is responsible for initiating the reconfiguration to a new image with the dedicated circuitry. Application images implement user-defined functionality in the target Stratix V device. You may also include the default application image functionality in the factory image. Each Stratix V device in your system requires one factory image and one or more application images.

Remote Update Mode

Stratix V remote system upgrade circuitry only supports remote update mode. In remote update mode, Stratix V devices load the factory configuration image after power up. The user-defined factory configuration determines which application configuration is to be loaded and triggers a reconfiguration cycle.

When a Stratix V device is first powered up in remote update mode, it loads the factory configuration located at the start address of `PGM[23..0]` = 24'h000000 in the EPCS and EPCQ devices. You must store your factory configuration image at this start address.

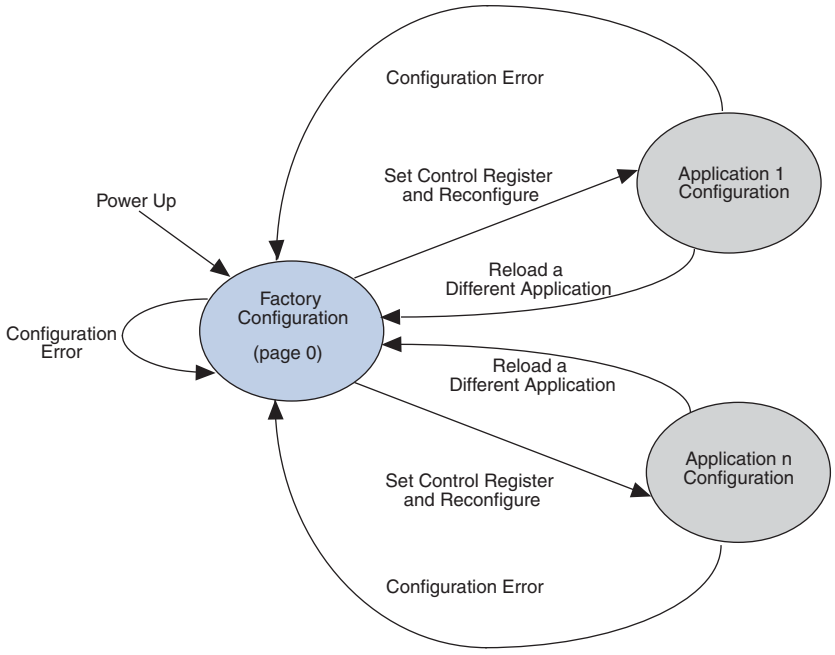
The application image start address can be at any EPCS or EPCQ sector boundary. Altera recommends using different sectors in the EPCS device for two images.

The factory image is user-designed and contains soft logic to perform the following:

- Process any errors based on status information from the dedicated remote system upgrade circuitry
- Communicate with the remote host and receive new application configurations and store this new configuration data in the local non-volatile memory device
- Determine which application configuration is to be loaded into the Stratix V device
- Enable or disable the user watchdog timer and load its time-out value
- Instruct the dedicated remote system upgrade circuitry to start a reconfiguration cycle

Figure 9–28 shows the transitions between the factory and application configurations in remote update mode.

Figure 9–28. Transitions Between Configurations in Remote Update Mode



After power up or a configuration error, the factory configuration image is loaded automatically. The system then decides to switch to the application configuration image or to stay in the factory configuration image. After the system decides to switch to an application configuration image, a reconfiguration is initiated through the remote system upgrade circuitry. In the application configuration image, the system may revert back to factory configuration image after the following reconfiguration trigger conditions are met:

- nSTATUS driven low externally
- Configuration CRC error
- User watchdog timer time-out
- Core nCONFIG signal assertion
- External nCONFIG signal assertion

After the factory configuration image is re-loaded, the user-designed factory configuration can read the remote system upgrade status register to determine the reason for the reconfiguration. The factory configuration then takes the appropriate error recovery steps and writes to the remote system upgrade control register to determine the next application configuration to be loaded.

Whenever the application configuration image is successfully loaded, the soft logic (Nios II processor or state machine and the remote communication interface) determine when the remote system update is arriving. When this occurs, the soft logic receives the incoming data, writes it to the configuration memory device, and triggers the device to load the factory configuration. The factory configuration reads the remote system upgrade status register and control register, determines the valid application configuration to load, writes the remote system upgrade control register accordingly, and initiates system reconfiguration.

Remote System Upgrade Using EPCQ 256

When you are using EPCQ 256, ensure that the application image address granularity is 32'h00000100. The **.rbf** size for your application image is 76,500 bytes longer than the numbers listed in Table 9–5 on page 9–8. You need to take this extra space requirement into consideration when you try to fit multiple application images in the EPCQ 256 device.

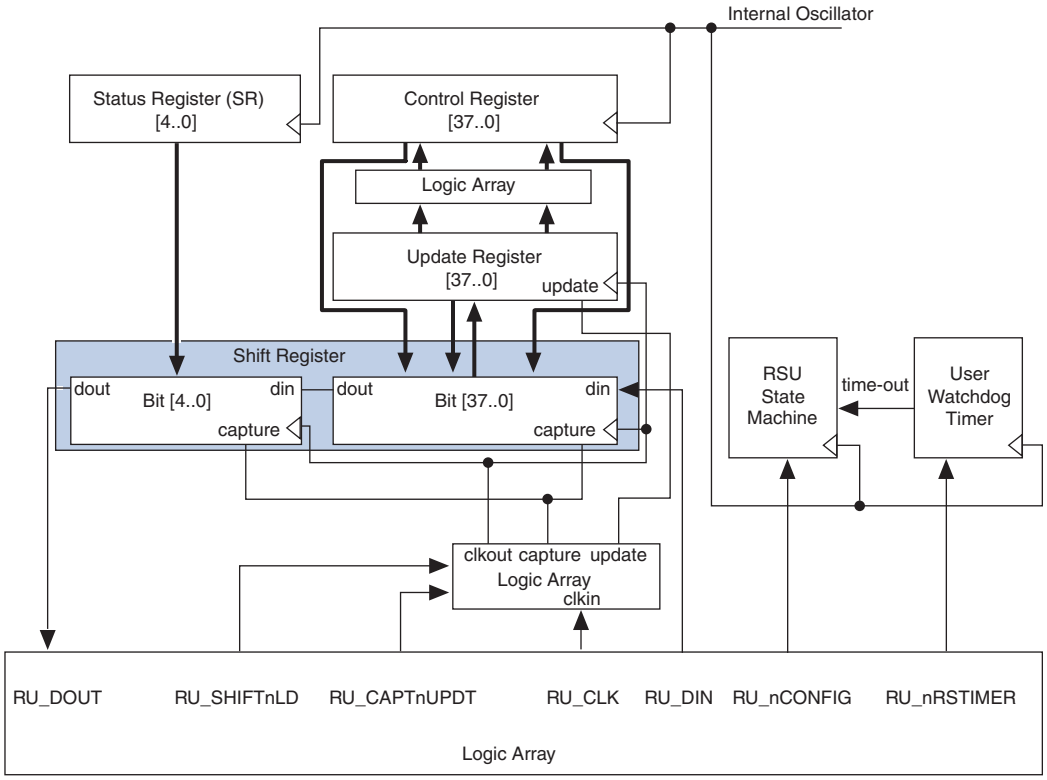
If you are not using Quartus II software or SRunner software for EPCQ 256 programming, put your EPCQ 256 device into four-byte addressing mode before you program and configure your device.

Dedicated Remote System Upgrade Circuitry

This section describes the implementation of the Stratix V dedicated remote system upgrades circuitry. The remote system upgrade circuitry is implemented in hard logic. This dedicated circuitry interfaces with the user-defined factory and application configurations implemented in the Stratix V device logic array to provide the complete remote configuration solution. The remote system upgrade circuitry contains the remote system upgrade registers, a watchdog timer, and a state machine that controls those components.

Figure 9–29 shows the remote system upgrade circuitry.

Figure 9–29. Remote System Upgrade Circuitry ***(Note 1)***



Note to Figure 9–29:

(1) If you are using the ALTREMOTE_UPDATE megafunction, the `RU_DOUT`, `RU_SHIFTnLD`, `RU_CAPTnUPDT`, `RU_CLK`, `RU_DIN`, `RU_nCONFIG`, and `RU_nRSTIMER` signals are internally controlled by the megafunction to perform all the related remote system upgrade operations.

Table 9–15 lists the timing parameter specifications for the remote system upgrade circuitry.

Table 9–15. Remote System Upgrade Circuitry Timing Specifications

Parameter	Minimum	Maximum	Unit
$f_{MAX\_RU\_CLK}$ *(1)*	—	40	MHz
$t_{RU\_nCONFIG}$ *(2)*	250	—	ns
$t_{RU\_nRSTIMER}$ *(3)*	250	—	ns

Notes to Table 9–15:

(1) This clock is user-supplied to the remote system upgrade circuitry. If you are using the ALTREMOTE_UPDATE megafunction, the clock user-supplied to the ALTREMOTE_UPDATE megafunction must meet this specification.

(2) This is equivalent to strobing the reconfiguration input of the ALTREMOTE_UPDATE megafunction high for the minimum timing specification. For more information, refer to "Remote System Upgrade State Machine" on page 9–51.

(3) This is equivalent to strobing the reset_timer input of the ALTREMOTE_UPDATE megafunction high for the minimum timing specification. For more information, refer to "User Watchdog Timer" on page 9–51.

Remote System Upgrade Registers

The remote system upgrade block contains a series of registers that store the page addresses, watchdog timer settings, and status information. Table 9–16 lists these registers.

Table 9–16. Remote System Upgrade Registers

Register	Description
Shift register	This register is accessible by the logic array and allows the update, status, and control registers to be written and sampled by user logic.
Control register	This register contains the current page address, user watchdog timer settings, and one bit specifying whether the current configuration is a factory configuration or an application configuration. During a read operation in an application configuration, this register is read into the shift register. When a reconfiguration cycle is initiated, the contents of the update register are written into the control register.
Update register	This register contains data similar to that in the control register. However, it can only be updated by the factory configuration by shifting data into the shift register and issuing an update operation. When a reconfiguration cycle is triggered by the factory configuration, the control register is updated with the contents of the update register. During a capture in a factory configuration, this register is read into the shift register.
Status register	This register is written to by the remote system upgrade circuitry on every reconfiguration to record the cause of the reconfiguration. This information is used by the factory configuration to determine the appropriate action following a reconfiguration. During a capture cycle, this register is read into the shift register.

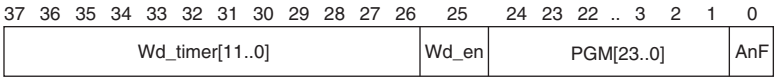
The remote system upgrade control and status registers are clocked by the 10-MHz internal oscillator (the same oscillator that controls the user watchdog timer). However, the remote system upgrade shift and update registers are clocked by the user clock input (RU_CLK).

Control Register

The control register stores the application configuration page address and user watchdog timer settings. The control register functionality depends on the remote system upgrade mode selection. A factory configuration in remote update mode has write access to this register.

Figure 9–30 shows the control register bit positions. Table 9–17 lists the control register bits. In the figure, the numbers show the bit position of a setting in a register. For example, bit number 25 is the enable bit for the watchdog timer.

Figure 9–30. Remote System Upgrade Control Register



The application-not-factory (AnF) bit indicates whether the current configuration loaded in the Stratix V device is the factory configuration or an application configuration. This bit is set low by the remote system upgrade circuitry when an error condition causes a fall-back to the factory configuration. When the AnF bit is high, the control register restricts the access to only read operations and enables the watchdog timer. The factory configuration design must set this bit high (1'b1) when updating the contents of the update register with the application page address and watchdog timer settings.

Table 9–17 lists the remote system upgrade control register contents.

Table 9–17. Remote System Upgrade Control Register Contents

Control Register Bit	Value *(2)*	Definition
AnF *(1)*	1'b0	Application not factory
PGM[23..0]	24'b0×000000	AS configuration start address (StAdd[23..0])
Wd_en	1'b0	User watchdog timer enable bit
Wd_timer[11..0]	12'b000000000000	User watchdog time-out value (most significant 12 bits of 29-bit count value: {Wd_timer[11..0], 17'b0})

Notes to Table 9–17:

(1) Factory configuration designs must set the AnF bit to **1'b1** before triggering the reconfiguration to application configuration image.

(2) This is the default value of the control register bit after the device exits POR and during reconfiguration back to the factory configuration image after reconfiguration trigger conditions.

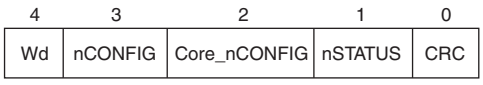
Status Register

The status register specifies the reconfiguration trigger condition. Figure 9–31 shows the status register content. The following list defines each bit:

- Bit 0—CRC error during application configuration
- Bit 1—nSTATUS assertion by an external device due to an error
- Bit 2—Stratix V device logic array triggered a reconfiguration cycle, possibly after downloading a new application configuration image
- Bit 3—external configuration reset (nCONFIG) assertion
- Bit 4—user watchdog timer time-out

Figure 9–31 shows the contents of the status register. The numbers in the figure show the bit positions in a 5-bit register.

Figure 9–31. Remote System Upgrade Status Register *(Note 1)*



Note to Figure 9–31:

(1) After the device exits POR and powers-up, the status register content is 5'b00000.

Remote System Upgrade State Machine

After power-up, the shift register, control register, and update registers are reset to the values listed in Table 9–16, also known as POR reset values before the factory configuration image is loaded. In the factory configuration image, the user logic writes the AnF bit, page address, and watchdog timer settings for the next application configuration image to the update register. When the logic array configuration reset (RU_nCONFIG) goes low, the remote system upgrade state machine updates the control register with the contents of the update register, and triggers a reconfiguration to the new application configuration image.

If there is an error during reconfiguration to the new application configuration image, the remote system upgrade state machine directs the system to re-load a factory configuration image. The control and update registers are reset to POR reset values and the status register is updated with the error information. For example, if there is a CRC error during application configuration image configuration, the status register is updated with 5'b00001.

If there is no error during reconfiguration and the application configuration image is successfully loaded, the system stays in the application configuration image until another reconfiguration trigger condition occurs. This can be a core nCONFIG assertion, external nCONFIG assertion, or the watchdog timer time-out error. If this happens, the control register and update registers are reset to POR reset values and the status register is updated with the error information. Consequently, the system proceeds to load the factory configuration image. Based on the status register content, the user logic in the factory configuration image then decides to stay in the factory configuration image or reload a new application reconfiguration image.

Read operations during factory configuration access the contents of the update register. This feature is used by the factory configuration image user logic to verify that the page address and watchdog timer settings are written correctly. Read operations in application configurations access the contents of the control register. This information is used by the user logic in the application configuration.

User Watchdog Timer

The user watchdog timer prevents a faulty application configuration from stalling the device indefinitely. The system uses the timer to detect functional errors after an application configuration is successfully loaded into the Stratix V device. This feature is automatically disabled in the factory configuration image and enabled in the application configuration image. Functional errors must not exist in the factory configuration because they are stored and validated during production and must never be updated remotely.

The user watchdog timer feature is automatically enabled in the application configuration image. If you do not wish to use this feature, disable it during the factory configuration image operation before triggering the reconfiguration to the application configuration image.

The user watchdog timer is a counter that counts down from the initial value loaded into the remote system upgrade control register by the factory configuration. The counter is 29 bits wide and has a maximum count value of 2^{29}. When specifying the user watchdog timer value, specify only the most significant 12 bits. The granularity of the timer setting is 2^{17} cycles. The cycle time is based on the frequency of the 12.5-MHz internal oscillator. Table 9–18 lists the operating range of the 12.5-MHz internal oscillator.

Table 9–18. 12.5-MHz Internal Oscillator Specifications ***(Note 1)***

Minimum	Typical	Maximum	Units
5.3	7.9	12.5	MHz

Note to Table 9–18:

(1) These values are preliminary.

The user watchdog timer begins counting after the application configuration enters device user mode. This timer must be periodically reset by the application configuration before the timer expires by asserting `RU_nRSTIMER`. If the application configuration does not reload the user watchdog timer before the count expires, a time-out signal is generated by the remote system upgrade dedicated circuitry. This causes the device to reload the factory configuration image and update the status register to reflect the watchdog timer time-out error.

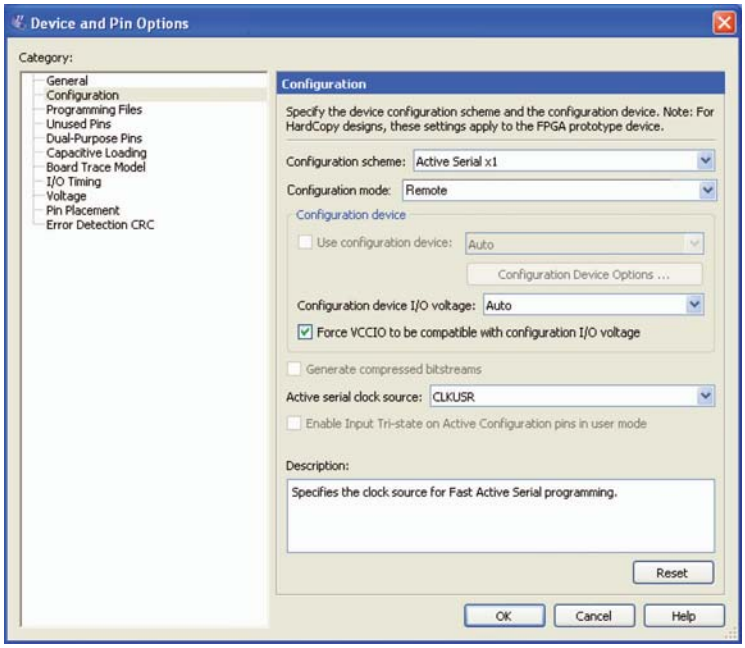
Enabling the Remote System Update Feature

You can enable remote update for Stratix V devices in the Quartus II software before design compilation (in the Compiler Settings menu). In remote update mode, the **auto-restart configuration after error** option is always enabled. To enable remote update in the project's compiler settings in the Quartus II software, follow these steps:

1. On the Assignments menu, click **Device**. The **Settings** dialog box appears.
2. Click **Device and Pin Options**. The **Device and Pin Options** dialog box appears.
3. Click the **Configuration** panel.
4. From the Configuration scheme list, select **Active Serial x1** (you can also use **Configuration Device**) (refer to Figure 9–32).
5. From the Configuration mode list, select **Remote** (refer to Figure 9–32).
6. Click **OK**.

7. In the **Settings** dialog box, click **OK**.

Figure 9–32. Enabling Remote Update for Stratix V Devices in the Compiler Settings Menu



ALTREMOTE_UPDATE Megafunction

The ALTREMOTE_UPDATE megafunction provides a memory-like interface to the remote system upgrade circuitry and handles the shift register read and write protocol in the Stratix V device logic. This implementation is suitable for designs that implement the factory configuration functions using a Nios II processor or user logic in the device. Using the megafunction block instead of creating your own logic saves design time and offers more efficient logic synthesis and device implementation.

For more information about the ALTREMOTE_UPDATE megafunction, refer to the *Remote Update Circuitry (ALTREMOTE_UPDATE) Megafunction User Guide*.

Design Security

This section provides an overview of the design security features and their implementation in Stratix V devices using the advanced encryption standard (AES). It also describes the security modes available in Stratix V devices that allow you to use these new features in your designs.

As Stratix V devices continue to play roles in larger and more critical designs in competitive commercial and military environments, it is increasingly important to protect your designs from copying, reverse engineering, and tampering. Stratix V design security supports the following features:

- Enhanced built-in AES decryption block to support 256-key industry-standard design security algorithm (FIPS-197 Certified)
- Volatile and non-volatile key programming support

- Secure operation mode for both volatile and non-volatile key through tamper protection bit setting
- JTAG secure mode is enable through tamper-protection bit
- Supports board level testing
- Supports in-socket key programming for non-volatile key
- Available in all configuration schemes except JTAG
- Supports both remote system upgrades and decompression feature

You can use the design security feature with or without the remote system upgrades or decompression features.

The Stratix V design security feature provides the following security protection for your designs:

- Security against copying—the security key is securely stored in the Stratix V device and cannot be read out through any interface. In addition, as configuration file read-back is not supported in Stratix V devices, your design information cannot be copied.
- Security against reverse engineering—reverse engineering from an encrypted configuration file is very difficult and time consuming because the Stratix V configuration file formats are proprietary and the file contains millions of bits that require specific decryption. In addition, the Stratix V devices are manufactured on the most advanced 28-nm process technology, making this process very difficult.
- Security against tampering—this disables tamper attempts through the JTAG interface. You can enhance this security feature with the tamper protection bit setting. After the tamper protection bit is set, the Stratix V device can only accept configuration files encrypted with the same key. Additionally, programming through the JTAG interface is blocked. This prevents any attempts to tamper with the device from both the JTAG interface and the configuration interface.

When you use compression with the design security feature, the configuration file is first compressed and then encrypted using the Quartus II software. During configuration, the Stratix V device first decrypts and then decompresses the configuration file.

When you use design security with Stratix V devices in a FPP configuration scheme, it requires a different DCLK-to-DATA[] ratio. For more information, refer to "Fast Passive Parallel Configuration" on page 9–9.

JTAG Secure Mode

When you enable tamper-protection bit, Stratix V devices are in JTAG secure mode after power-up. During JTAG secure mode, many JTAG instructions are disabled. Stratix V devices only allow mandatory JTAG 1149.1 and 1149.6 instructions to be exercised. These instructions are SAMPLE/PRELOAD, BYPASS, EXTEST, and optional instructions such as IDCODE and SHIFT_EDERROR_REG.

To enable the access of other JTAG instructions such as USERCODE, HIGHZ, CLAMP, PULSE_NCONFIG, and CONFIG_IO, you must issue UNLOCK instruction to deactivate the JTAG secure mode. You can issue LOCK instruction to put the device back into JTAG secure mode. Both the LOCK and UNLOCK instructions can only be issued during user mode.

For more information about JTAG binary instruction code related to the LOCK and UNLOCK instructions, refer to the *JTAG Boundary-Scan Testing in Stratix V Devices* chapter.

Security Key Types

Stratix V devices offer two types of keys—volatile and non-volatile. Table 9–19 lists the differences between the volatile key and non-volatile key.

Table 9–19. Security Key Types

Key Types	Key Programmability	Power Supply for Key Storage	Programming Method
Volatile Key	■ Re-programmable ■ Erasable	Required external battery, V_{CCBAT} *(1)*	On-board
Non-volatile Key	One-time programming	Does not require an external battery	On-board and in-socket programming *(2)*

Notes to Table 9–19:

(1) V_{CCBAT} is a dedicated power supply for volatile key storage and not shared with other on-chip power supplies, such as V_{CCIO} or V_{CCPGM}. V_{CCBAT} continuously supplies power to the volatile register regardless of the on-chip supply condition.

(2) In-socket programming is offered through third party vendors.

Both non-volatile and volatile key programming offers protection from reverse engineering and copying. If you set the tamper-protection bit, the design is also protected from tampering.

Perform key programming through the JTAG interface. Also, ensure that the nSTATUS pin is released high before any key-programming attempts.

For more information about battery specifications, refer to the *DC and Switching Characteristics for Stratix V Devices* chapter.

For more information about the V_{CCBAT} pin connection recommendations, refer to the *Stratix V Device Family Pin Connection Guidelines*.

Security Modes

Table 9–20 lists the security modes available in Stratix V devices.

Table 9–20. Supported Security Modes

Security Mode	Tamper protection Bit Setting	Device Accepts Unencrypted File	Device Accepts Encrypted File	Security Level
No-key	—	Yes	No	—
Volatile Key	—	Yes *(2)*	Yes	Secure
Volatile Key with Tamper Protection Bit Set	Set *(1)*	No	Yes	Secure with tamper resistant
Non-volatile Key	—	Yes *(2)*	Yes	Secure
Non-volatile Key with Tamper Protection Bit Set	Set *(1)*	No	Yes	Secure with tamper resistant

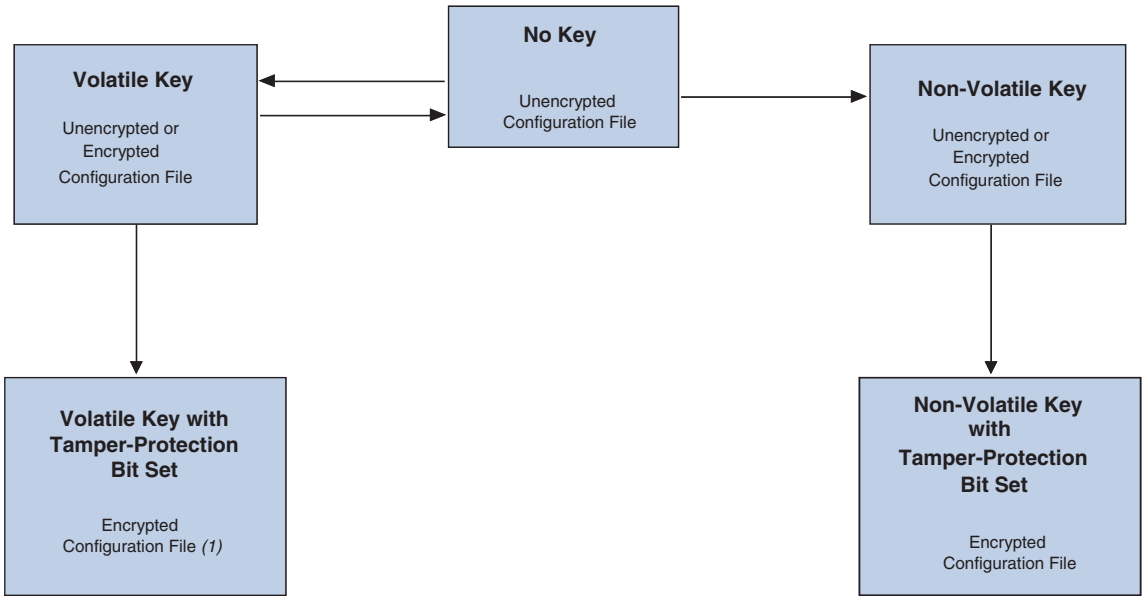
Notes to Table 9–20:

(1) Enabling the tamper protection bit disables test mode in Stratix V devices and disables programming through the JTAG interface. This process is irreversible and prevents Altera from carry-out failure analysis. Contact Altera Technical Support to enable the tamper protection bit.

(2) Use the unencrypted configuration bitstream support only for board-level testing.

Figure 9–33 shows the sequence of the security modes available in Stratix V devices.

Figure 9–33. Stratix V Security Modes—Sequence and Restrictions



Note to Figure 9–33:

(1) Stratix V devices do not accept the encrypted configuration file if the volatile key is erased. You must use the volatile key without tamper-protection bit set to reprogram the key if the volatile key in Stratix V device is erased.

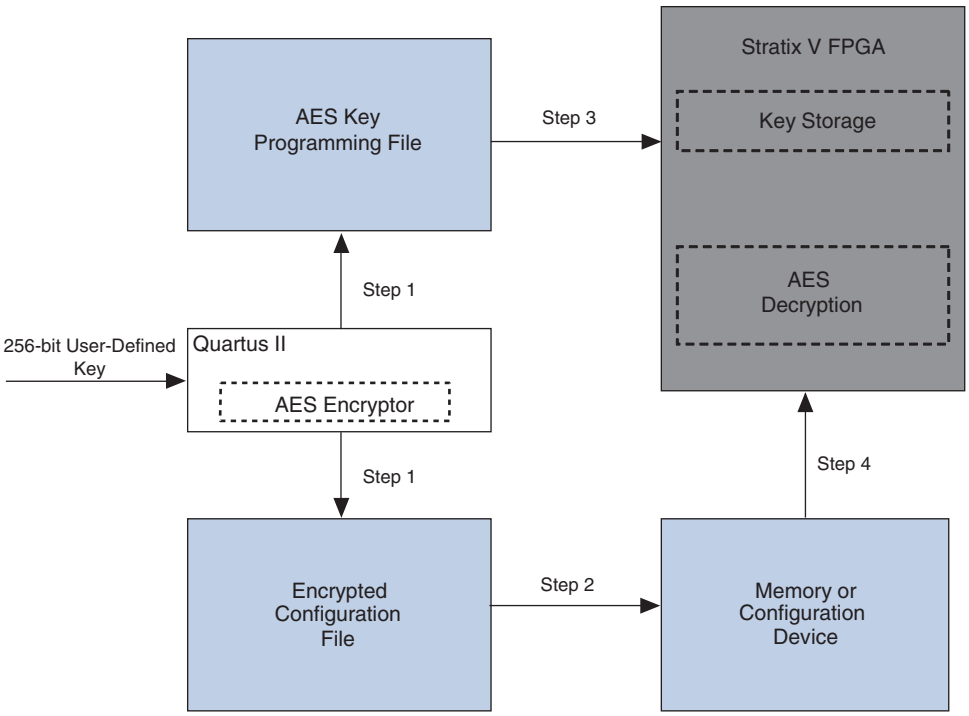
Design Security Implementation Steps

Stratix V devices are SRAM-based devices. To provide design security, Stratix V devices require a 256-bit security key for configuration bitstream design security. To carry out secure configuration, follow these steps:

1. The Quartus II software generates the design security key programming file and encrypts the configuration data using the user-defined 256-bit key.
2. Store the encrypted configuration file in the external memory.
3. Program the AES key programming file into the Stratix V device through a JTAG interface.
4. Configure the Stratix V device. At system power-up, the external memory device sends the encrypted configuration file to the Stratix V device.

Figure 9–34 shows the design security implementation steps.

Figure 9–34. Design Security Implementation Steps



Document Revision History

Table 9–21 lists the revision history for this chapter.

Table 9–21. Document Revision History

Date	Version	Changes
May 2011	1.3	■ Chapter moved to volume 2 for the 11.0 release. ■ Added "Remote System Upgrade Using EPCQ 256" and "JTAG Secure Mode" sections. ■ Updated Table 9–5. ■ Updated "Configuration", "Configuration Error", "Programming EPCS and EPCQ", "JTAG Configuration", "Remote Update Mode", and "Design Security" sections. ■ Minor text edits.
January 2011	1.2	■ Updated Table 9–7, Table 9–8, Table 9–12, and Table 9–14. ■ Updated Figure 9–15 and Figure 9–21. ■ Updated "User Watchdog Timer", "DCLK-to-DATA[] Ratio for FPP configuration", "V_{CCPD} Pin", "POR Delay Specification", and "Programming EPCS and EPCQ" sections.
December 2010	1.1	No changes to the content of this chapter for the Quartus II software 10.1.
July 2010	1.0	Initial release.

10. Configuration, Design Security, and Remote System Upgrades in Stratix IV Devices

SIV51010-3.3

This chapter describes the configuration, design security, and remote system upgrades in Stratix® IV devices. To save configuration memory space and time, Stratix IV devices provide configuration data decompression. They also provide a built-in design security feature that protects your designs against IP theft and tampering of your configuration files.

Stratix IV devices also offer remote system upgrade capability so that you can upgrade your system in real-time through any network. This helps to deliver feature enhancements and bug fixes and provides error detection, recovery, and status information to ensure reliable reconfiguration.

Overview

This chapter describes supported configuration schemes for Stratix IV devices, instructions about how to execute the required configuration schemes, and the necessary pin settings.

Stratix IV devices use SRAM cells to store configuration data. As SRAM is volatile, you must download configuration data to the Stratix IV device each time the device powers up. You can configure Stratix IV devices using one of four configuration schemes:

- Fast passive parallel (FPP)
- Fast active serial (AS)
- Passive serial (PS)
- Joint Test Action Group (JTAG)

All configuration schemes use either an external controller (for example, a MAX® II device or microprocessor), a configuration device, or a download cable. For more information, refer to "Configuration Features" on page 10–4.

This chapter includes the following sections:

- "Configuration Schemes" on page 10–2
- "Configuration Features" on page 10–4
- "Fast Passive Parallel Configuration" on page 10–6
- "Fast Active Serial Configuration (Serial Configuration Devices)" on page 10–16
- "Passive Serial Configuration" on page 10–24
- "JTAG Configuration" on page 10–34
- "Device Configuration Pins" on page 10–39
- "Configuration Data Decompression" on page 10–47
- "Remote System Upgrades" on page 10–49

© 2011 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

- "Remote System Upgrade Mode" on page 10–53
- "Dedicated Remote System Upgrade Circuitry" on page 10–56
- "Quartus II Software Support" on page 10–62
- "Design Security" on page 10–63

Configuration Devices

Altera® serial configuration devices support a single-device and multi-device configuration solution for Stratix IV devices and are used in the fast AS configuration scheme. Serial configuration devices offer a low-cost, low pin-count configuration solution.

For information about serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook*.

All minimum timing information in this chapter covers the entire Stratix IV family. Some devices may work at less than the minimum timing stated in this handbook due to process variation.

Configuration Schemes

Select the configuration scheme by driving the Stratix IV device MSEL pins either high or low, as shown in Table 10–1. The MSEL input buffers are powered by the V_{CC} power supply. Altera recommends hard wiring the MSEL[] pins to V_{CCPGM} and GND. The MSEL[2..0] pins have 5-kΩ internal pull-down resistors that are always active. During power-on reset (POR) and during reconfiguration, the MSEL pins must be at V_{IL} and V_{IH} levels of V_{CCPGM} voltage to be considered logic low and logic high.

To avoid problems with detecting an incorrect configuration scheme, hardwire the MSEL[] pins to V_{CCPGM} and GND without pull-up or pull-down resistors. Do not drive the MSEL[] pins by a microprocessor or another device.

Table 10–1. Configuration Schemes for Stratix IV Devices (Part 1 of 2)

Configuration Scheme	MSEL2	MSEL1	MSEL0
Fast passive parallel	0	0	0
Passive serial	0	1	0
Fast AS (40 MHz) *(1)*	0	1	1
Remote system upgrade fast AS (40 MHz) *(1)*	0	1	1
FPP with design security feature and/or decompression enabled *(2)*	0	0	1

Table 10–1. Configuration Schemes for Stratix IV Devices (Part 2 of 2)

Configuration Scheme	MSEL2	MSEL1	MSEL0
JTAG-based configuration *(4)*	*(3)*	*(3)*	*(3)*

Notes to Table 10–1:

(1) Stratix IV devices only support fast AS configuration. You must use either EPCS64 or EPCS128 devices to configure a Stratix IV device in fast AS mode.

(2) These modes are only supported when using a MAX II device or a microprocessor with flash memory for configuration. In these modes, the host system must output a DCLK that is ×4 the data rate.

(3) Do not leave the MSEL pins floating, connect them to V_{CCPGM} or GND. These pins support the non-JTAG configuration scheme used in production. If you only use the JTAG configuration, connect the MSEL pins to GND.

(4) The JTAG-based configuration takes precedence over other configuration schemes, which means the MSEL pin settings are ignored. The JTAG-based configuration does not support the design security or decompression features.

Table 10–2 lists the uncompressed raw binary file (**.rbf**) configuration file sizes for Stratix IV devices.

Table 10–2. Uncompressed Raw Binary File (.rbf) Sizes for Stratix IV Devices

Device	Data Size (Bits)
EP4SE230	94,557,465
EP4SE360	128,395,577
EP4SE530	171,722,057
EP4SE820	241,684,465
EP4SGX70	47,833,345
EP4SGX110	47,833,345
EP4SGX180	94,557,465
EP4SGX230	94,557,465
EP4SGX290	128,395,577
	171,722,057 *(1)*
EP4SGX360	128,395,577
	171,722,057 *(1)*
EP4SGX530	171,722,057
EP4S40G2	94,557,465
EP4S40G5	171,722,057
EP4S100G2	94,557,465
EP4S100G3	171,722,057
EP4S100G4	171,722,057
EP4S100G5	171,722,057

Note to Table 10–2:

(1) This only applies to the F45 package.

Use the data in Table 10–2 to estimate the file size before design compilation. Different configuration file formats; for example, a hexidecimal (**.hex**) or tabular text file (**.ttf**) format, have different file sizes. Refer to the Quartus® II software for the different types of configuration file and file sizes. However, for any specific version of the Quartus II software, any design targeted for the same device has the same uncompressed configuration file size. If you are using compression, the file size can vary after each compilation because the compression ratio depends on the design.

For more information about setting device configuration options or creating configuration files, refer to the *Device Configuration Options* and *Configuration File Formats* chapters in volume 2 of the *Configuration Handbook*.

Configuration Features

Stratix IV devices offer design security, decompression, and remote system upgrade features. Design security using configuration bitstream encryption is available in Stratix IV devices, which protects your designs. Stratix IV devices can receive a compressed configuration bitstream and decompress this data in real-time, reducing storage requirements and configuration time. You can make real-time system upgrades from remote locations of your Stratix IV designs with the remote system upgrade feature.

Table 10–3 lists which configuration features you can use in each configuration scheme.

Table 10–3. Configuration Features for Stratix IV Devices

Configuration Scheme	Configuration Method	Decompression	Design Security	Remote System Upgrade
FPP	MAX II device or a microprocessor with flash memory	✓ *(1)*	✓ *(1)*	—
Fast AS	Serial configuration device	✓	✓	✓
PS	MAX II device or a microprocessor with flash memory	✓	✓	—
	Download cable	✓	✓	—
JTAG	MAX II device or a microprocessor with flash memory	—	—	—
	Download cable	—	—	—

Note to Table 10–3:

(1) In these modes, the host system must send a DCLK that is ×4 the data rate.

You can also refer to the following:

- For more information about the configuration data decompression feature, refer to "Configuration Data Decompression" on page 10–47.
- For more information about the remote system upgrade feature, refer to "Remote System Upgrades" on page 10–49.
- For more information about the design security feature, refer to "Design Security" on page 10–63.

If your system already contains a common flash interface (CFI) flash memory, you can use it for Stratix IV device configuration storage as well. The MAX II parallel flash loader (PFL) feature in MAX II devices provides an efficient method to program CFI flash memory devices through the JTAG interface and provides the logic to control configuration from the flash memory device to the Stratix IV device. Both PS and FPP configuration modes are supported using this PFL feature.

For more information about PFL, refer to *Parallel Flash Loader Megafunction User Guide*.

For more information about programming Altera serial configuration devices, refer to "Programming Serial Configuration Devices" on page 10–22.

Power-On Reset Circuit

The POR circuit keeps the entire system in reset until the power supply voltage levels have stabilized on power-up. After power-up, the device does not release nSTATUS until V_{CC}, V_{CCAUX}, V_{CCPT}, V_{CCPGM}, and V_{CCPD} are above the device's POR trip point. On power down, brown-out occurs if the V_{CC}, V_{CCAUX}, V_{CCPT}, V_{CCPGM}, or V_{CCPD} drops below the threshold voltage.

In Stratix IV devices, a pin-selectable option (PORSEL) is provided that allows you to select between the standard POR time or fast POR time. When PORSEL is driven low, the standard POR time is 100 ms < T_{POR} < 300 ms, which has a lower power-ramp rate. When PORSEL is driven high, the fast POR time is 4 ms < T_{POR} < 12 ms.

V_{CCPGM} Pins

Stratix IV devices have a power supply, V_{CCPGM}, for all the dedicated configuration pins and dual function pins. The supported configuration voltage is 1.8, 2.5, and 3.0 V. Stratix IV devices do not support 1.5 V configuration.

Use the V_{CCPGM} pin to power all dedicated configuration inputs, dedicated configuration outputs, dedicated configuration bidirectional pins, and some of the dual functional pins that you use for configuration. With V_{CCPGM}, the configuration input buffers do not have to share power lines with the regular I/O buffer in Stratix IV devices.

The operating voltage for the configuration input pin is independent of the I/O banks power supply V_{CCIO} during configuration. Therefore, Stratix IV devices do not need configuration voltage constraints on V_{CCIO}.

V_{CCPD} Pins

Stratix IV devices have a dedicated programming power supply, V_{CCPD}, which must be connected to 3.0 V/2.5 V to power the I/O pre-drivers and JTAG I/O pins (TCK, TMS, TDI, TDO, and TRST).

V_{CCPGM} and V_{CCPD} must ramp up from 0 V to the desired voltage level within 100 ms when PORSEL is low or 4 ms when PORSEL is high. If these supplies are not ramped up within this specified time, your Stratix IV device will not configure successfully. If your system cannot ramp up the power supplies within 100 ms or 4 ms, you must hold nCONFIG low until all the power supplies are stable.

V_{CCPD} must be greater than or equal to V_{CCIO} of the same bank. If V_{CCIO} of the bank is set to 3.0 V, V_{CCPD} must be powered up to 3.0 V. If the V_{CCIO} of the bank is powered to 2.5 V or lower, V_{CCPD} must be powered up to 2.5 V.

For more information about configuration pins power supply, refer to "Device Configuration Pins" on page 10–39.

Fast Passive Parallel Configuration

Fast passive parallel configuration in Stratix IV devices is designed to meet the continuously increasing demand for faster configuration times. Stratix IV devices are designed with the capability of receiving byte-wide configuration data per clock cycle.

You can perform FPP configuration of Stratix IV devices using an intelligent host, such as a MAX II device or a microprocessor.

FPP Configuration Using a MAX II Device as an External Host

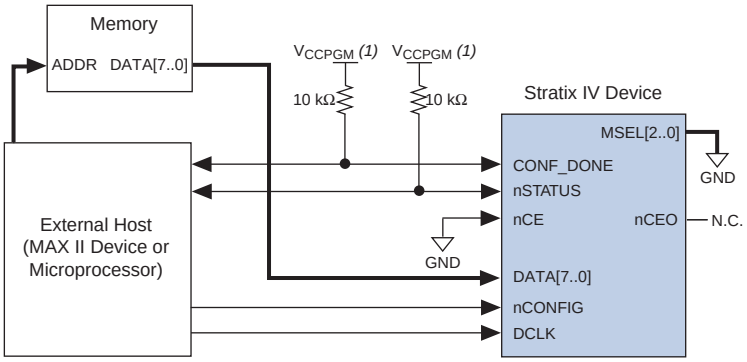
FPP configuration using compression and an external host provides the fastest method to configure Stratix IV devices. In this configuration scheme, you can use a MAX II device as an intelligent host that controls the transfer of configuration data from a storage device, such as flash memory, to the target Stratix IV device. You can store configuration data in **.rbf**, **.hex**, or **.ttf** format. When using the MAX II device as an intelligent host, a design that controls the configuration process, such as fetching the data from flash memory and sending it to the device, must be stored in the MAX II device.

If you are using the Stratix IV decompression and/or design security features, the external host must be able to send a DCLK frequency that is ×4 the data rate.

The ×4 DCLK signal does not require an additional pin and is sent on the DCLK pin. The maximum DCLK frequency is 125 MHz, which results in a maximum data rate of 250 Mbps. If you are not using the Stratix IV decompression or design security features, the data rate is ×8 the DCLK frequency.

Figure 10–1 shows the configuration interface connections between the Stratix IV device and a MAX II device for single device configuration.

Figure 10–1. Single Device FPP Configuration Using an External Host



Note to Figure 10–1:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Stratix IV device. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends powering up all configuration system I/Os with V_{CCPGM}.

After power-up, the Stratix IV device goes through a POR. The POR delay depends on the PORSEL pin setting. When PORSEL is driven low, the standard POR time is 100 ms < T_{POR} < 300 ms. When PORSEL is driven high, the fast POR time is 4 ms < T_{POR} < 12 ms. During POR, the device resets, holds nSTATUS low, and tri-states all user I/O pins. After the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power up and configuration, the user I/O pins and dual-purpose I/O pins have weak pull-up resistors, which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in the reset stage. To initiate configuration, the MAX II device must drive the nCONFIG pin from low to high.

To begin the configuration process, you must fully power V_{CCPT}, V_{CC}, V_{CCPD}, and V_{CCPGM} of the banks where the configuration pins reside to the appropriate voltage levels.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩpull-up resistor. After nSTATUS is released, the device is ready to receive configuration data and the configuration stage begins. When nSTATUS is pulled high, the MAX II device places the configuration data one byte at a time on the DATA[7..0] pins.

Stratix IV devices receive configuration data on the DATA[7..0] pins and the clock is received on the DCLK pin. Data is latched into the device on the rising edge of DCLK. If you are using the Stratix IV decompression and/or design security features, configuration data is latched on the rising edge of every fourth DCLK cycle. After the configuration data is latched in, it is processed during the following three DCLK cycles. Therefore, you can only stop DCLK after three clock cycles after the last data is latched into the Stratix IV devices.

Data is continuously clocked into the target device until CONF_DONE goes high. The CONF_DONE pin goes high one byte early in FPP modes. The last byte is required for FPP mode. After the device has received the next-to-last byte of the configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by an external 10-kΩ pull-up resistor. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin. The CONF_DONE pin must have an external 10-kΩ pull-up resistor for the device to initialize.

In Stratix IV devices, the initialization clock source is either the internal oscillator or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If you use the internal oscillator, the Stratix IV device provides itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device. Driving DCLK to the device after configuration is complete does not affect device operation.

You can also synchronize initialization of multiple devices or delay initialization with the CLKUSR option. You can turn on the **Enable user-supplied start-up clock (CLKUSR)** option in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box. Supplying a clock on CLKUSR does not affect the configuration process. The CONF_DONE pin goes high one byte early in FPP modes. The last byte is required for FPP mode. After the CONF_DONE pin transitions high, CLKUSR is enabled after the time specified at t_{CD2CU}. After this time period elapses, Stratix IV devices require 8,532 clock cycles to initialize properly and enter user mode. Stratix IV devices support a CLKUSR f_{MAX} of 125 MHz.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. This **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box. If you use the INIT_DONE pin, it is high because of an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. After the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. The MAX II device must be able to detect this low-to-high transition, which signals the device has entered user mode. When initialization is complete, the device enters user mode. In user-mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

Two DCLK falling edges are required after CONF_DONE goes high to begin the initialization of the device for both uncompressed and compressed bitstream in FPP.

To ensure DCLK and DATA[7..0] are not left floating at the end of configuration, the MAX II device must drive them either high or low, whichever is convenient on your board. The DATA[7..0] pins are available as user I/O pins after configuration. When you select the FPP scheme as a default in the Quartus II software, these I/O pins are tri-stated in user mode. To change this default option in the Quartus II software, select the **Dual-Purpose Pins** tab of the **Device and Pin Options** dialog box.

The configuration clock (DCLK) speed must be below the specified frequency to ensure correct configuration. No maximum DCLK period exists, which means you can pause configuration by halting DCLK for an indefinite amount of time.

If you need to stop DCLK, it can only be stopped:

- three clock cycles after the last data byte was latched into the Stratix IV device when you use the decompression and/or design security features.
- two clock cycles after the last data byte was latched into the Stratix IV device when you do not use the Stratix IV decompression and/or design security features.

By stopping DCLK, the configuration circuit allows enough clock cycles to process the last byte of latched configuration data. When the clock restarts, the MAX II device must provide data on the DATA[7..0] pins prior to sending the first DCLK rising edge.

If an error occurs during configuration, the device drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the MAX II device that there is an error. If the **Auto-restart configuration after error** option (available in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box) is turned on, the device releases nSTATUS after a reset time-out period (a maximum of 500 µs). After nSTATUS is released and pulled high by a pull-up resistor, the MAX II device can try to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

If you have enabled the **Auto-restart configuration after error** option, the nSTATUS pin transitions from high to low and back again to high when a configuration error is detected. This appears as a low pulse at the nSTATUS pin with a minimum pulse width of 10 µs to a maximum pulse width of 500 µs, as defined in the t_{STATUS} specification.

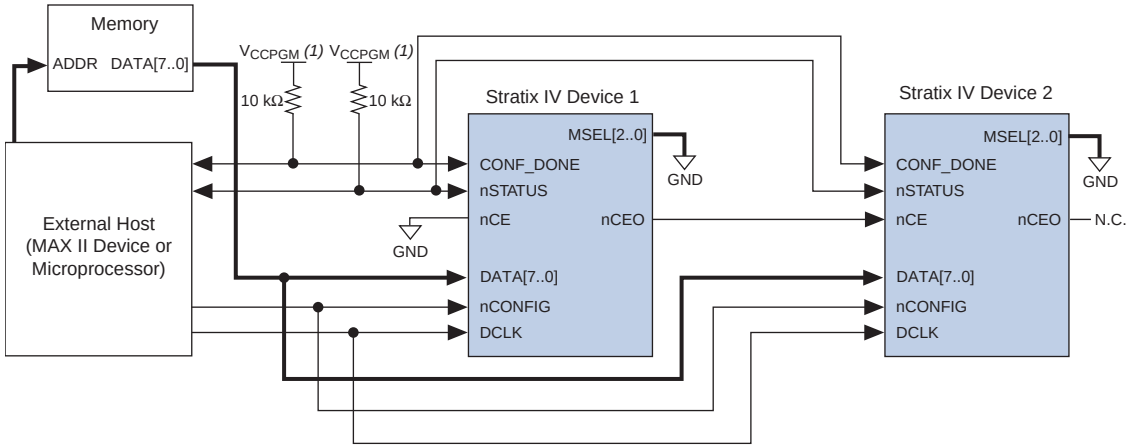
The MAX II device can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The MAX II device must monitor the CONF_DONE pin to detect errors and determine when programming completes. If all the configuration data is sent, but the CONF_DONE or INIT_DONE signals have not gone high, the MAX II device reconfigures the target device.

If you use the optional CLKUSR pin and nCONFIG is pulled low to restart the configuration during device initialization, ensure that CLKUSR continues toggling during the time nSTATUS is low (a maximum of 500 µs).

When the device is in user mode, initiating reconfiguration is done by transitioning the nCONFIG pin low-to-high. The nCONFIG pin must be low for at least 2 µs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. After nCONFIG returns to a logic high level and nSTATUS is released by the device, reconfiguration begins.

Figure 10–2 shows how to configure multiple Stratix IV devices using a MAX II device. This circuit is similar to the FPP configuration circuit for a single device, except the devices are cascaded for multi-device configuration.

Figure 10–2. Multi-Device FPP Configuration Using an External Host



Note to Figure 10–2:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for all Stratix IV devices in the chain. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O standard on the device and the external host. Altera recommends powering up all configuration system I/Os with V_{CCPGM}.

In a multi-device FPP configuration, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration within one clock cycle; therefore, the transfer of data destinations is transparent to the MAX II device. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. The configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

All nSTATUS and CONF_DONE pins are tied together; if any device detects an error, configuration stops for the entire chain and you must reconfigure the entire chain. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

If the **Auto-restart configuration after error** option is turned on, the devices release their nSTATUS pins after a reset time-out period (a maximum of 500 µs). After all nSTATUS pins are released and pulled high, the MAX II device tries to reconfigure the chain without pulsing nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

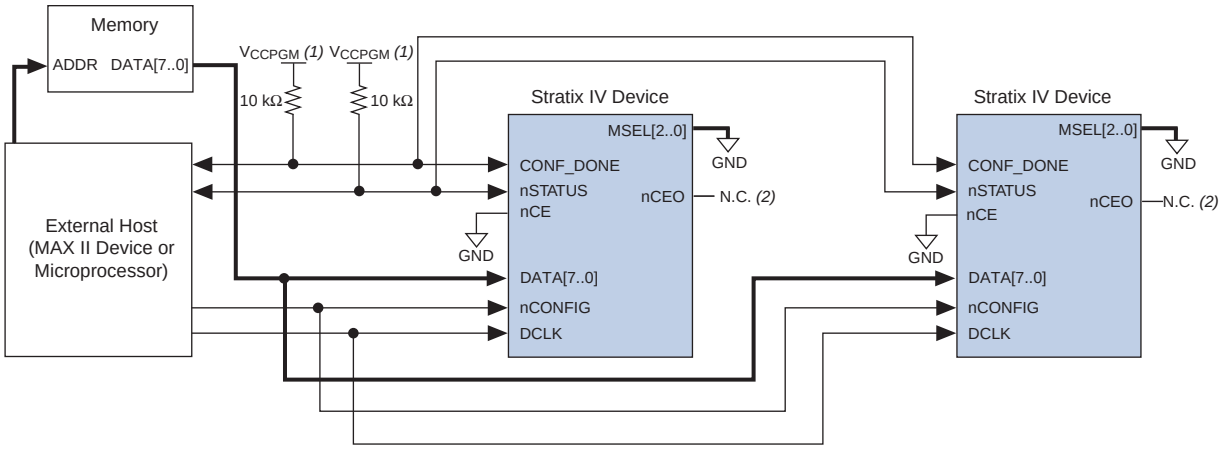
If you have enabled the **Auto-restart configuration after error** option, the nSTATUS pin transitions from high to low and back again to high when a configuration error is detected. This appears as a low pulse at the nSTATUS pin with a minimum pulse width of 10 µs to a maximum pulse width of 500 µs, as defined in the t_{STATUS} specification.

In a multi-device FPP configuration chain, all Stratix IV devices in the chain must either enable or disable the decompression and/or design security features. You cannot selectively enable the decompression and/or design security features for each device in the chain because of the DATA and DCLK relationship. If the chain contains devices that do not support design security, use a serial configuration scheme.

If a system has multiple devices that contain the same configuration data, tie all device nCE inputs to GND and leave the nCEO pins floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. Configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices start and complete configuration at the same time.

Figure 10–3 shows a multi-device FPP configuration when both Stratix IV devices are receiving the same configuration data.

Figure 10–3. Multiple-Device FPP Configuration Using an External Host When Both Devices Receive the Same Data



Notes to Figure 10–3:

(1) Connect the resistor to a supply that provides an acceptable input signal for all Stratix IV devices in the chain. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends powering up all configuration system I/Os with V_{CCPGM}.

(2) The nCEO pins of both Stratix IV devices are left unconnected when configuring the same configuration data into multiple devices.

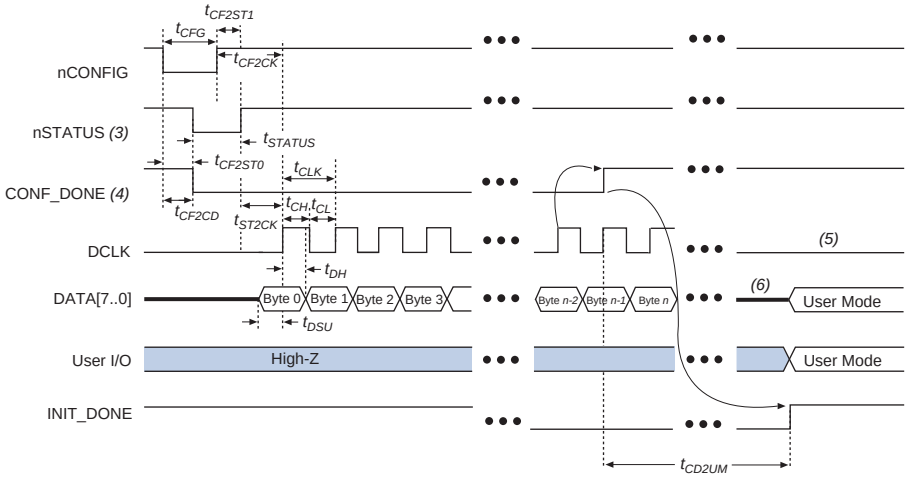
You can use a single configuration chain to configure Stratix IV devices with other Altera devices that support FPP configuration, such as other types of Stratix devices. To ensure that all devices in the chain complete configuration at the same time, or that an error flagged by one device initiates reconfiguration in all devices, tie all of the device CONF_DONE and nSTATUS pins together.

For more information about configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* in volume 2 of the *Configuration Handbook*.

FPP Configuration Timing

Figure 10–4 shows the timing waveform for an FPP configuration when using a MAX II device as an external host. This waveform shows the timing when you have not enabled the decompression and design security features.

Figure 10–4. FPP Configuration Timing Waveform *(Note 1)*, *(2)*



Notes to Figure 10–4:

(1) Use this timing waveform when you have not enabled the decompression and design security features.

(2) The beginning of this waveform shows the device in user mode. In user mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.

(3) After power-up, the Stratix IV device holds nSTATUS low for the time of the POR delay.

(4) After power-up, before and during configuration, CONF_DONE is low.

(5) Do not leave DCLK floating after configuration. You can drive it high or low, whichever is more convenient.

(6) DATA[7..0] are available as user I/O pins after configuration except for some exceptions on Stratix IV GT devices. The state of these pins depends on the dual-purpose pin settings.

Table 10–4 lists the timing parameters for Stratix IV devices for an FPP configuration when you have not enabled the decompression and design security features.

Table 10–4. FPP Timing Parameters for Stratix IV Devices (Part 1 of 2) *(Note 1)*, *(2)*

Symbol	Parameter	Minimum			Maximum			Units
		Stratix IV (7)	Stratix IV (8)	Stratix IV (9)	Stratix IV (7)	Stratix IV (8)	Stratix IV (9)	
t_{CF2CD}	nCONFIG low to CONF_DONE low	—			800			ns
t_{CF2ST0}	nCONFIG low to nSTATUS low	—			800			ns
t_{CFG}	nCONFIG low pulse width	2			—			µs
t_{STATUS}	nSTATUS low pulse width	10			500 (3)			µs
t_{CF2ST1}	nCONFIG high to nSTATUS high	—			500 (4)			µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	500			—			µs

Table 10–4. FPP Timing Parameters for Stratix IV Devices (Part 2 of 2) *(Note 1)*, *(2)*

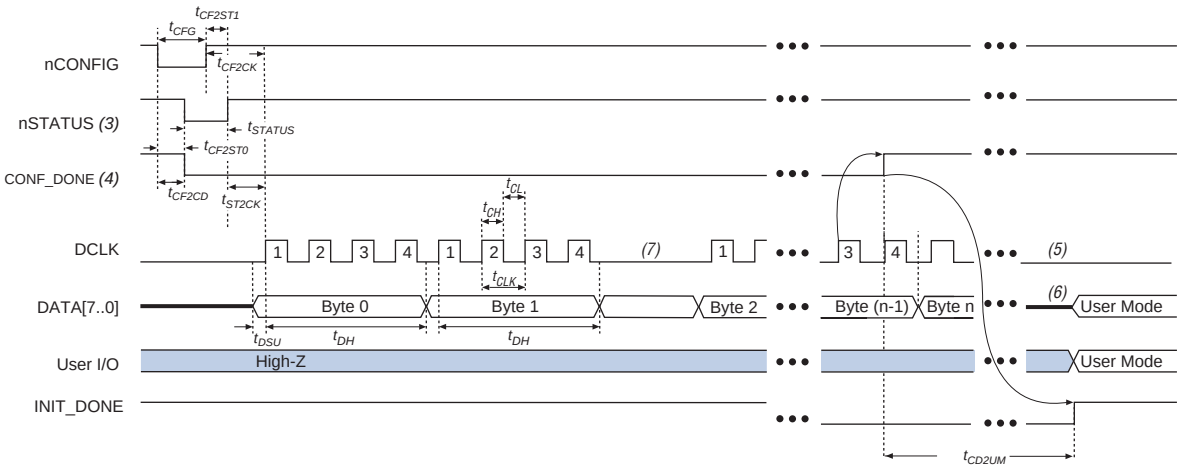
Symbol	Parameter	Minimum			Maximum			Units
		Stratix IV *(7)*	Stratix IV *(8)*	Stratix IV *(9)*	Stratix IV *(7)*	Stratix IV *(8)*	Stratix IV *(9)*	
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2			—			µs
t_{DSU}	Data setup time before rising edge on DCLK	4			—			ns
t_{DH}	Data hold time after rising edge on DCLK	1			—			ns
T_R	Input rise time	—			40			ns
t	Input fall time	—			40			ns
t_{CD2UM}	CONF_DONE high to user mode *(5)*	55			150			µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period			—			—
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	t_{CD2CU} + (8532 × CLKUSR period)			—			—
t_{CH}	DCLK high time *(6)*	3.6	4.5	5.6	—			ns
t_{CL}	DCLK low time *(6)*	3.6	4.5	5.6	—			ns
t_{CLK}	DCLK period *(6)*	8	10	12.5	—			ns
f_{MAX}	DCLK frequency	—			125	100	80	MHz

Notes to Table 10–4:

(1) This information is preliminary.
(2) Use these timing parameters when you have not enabled the decompression and design security features.
(3) You can obtain this value if you do not delay the configuration by extending the nCONFIG or nSTATUS low pulse width.
(4) This value is applicable if you do not delay the configuration by externally holding nSTATUS low.
(5) The minimum and maximum numbers apply only if you chose the internal oscillator as the clock source for starting the device.
(6) Adding up t_{CH} and t_{CL} equals to t_{CLK}. When EP4SE230 t_{CH} is 3.6 ns (min), t_{CL} must be 4.4 ns and vice versa.
(7) Applicable to EP4SE230, EP4SE360, EP4SGX70, EP4SGX110, EP4SGX180, EP4SGX230, EP4SGX290 (except F45 package), EP4SGX360 (except F45 package), EP4S40G2, EP4S100G2 devices.
(8) Applicable to EP4SE530, EP4SGX290 (only for F45 package), EP4SGX360 (only for F45 package), EP4SGX530, EP4S40G5, EP4S100G3, EP4S100G4, EP4S100G5 devices.
(9) Applicable to EP4SE820 only.

Figure 10–5 shows the timing waveform for an FPP configuration when using a MAX II device as an external host. This waveform shows the timing when you have enabled the decompression and/or design security features.

Figure 10–5. FPP Configuration Timing Waveform with Decompression or Design Security Feature Enabled *(Note 1), (2)*



Notes to Figure 10–5:

(1) Use this timing waveform when you have enabled the decompression and/or design security features.

(2) The beginning of this waveform shows the device in user-mode. In user-mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.

(3) After power-up, the Stratix IV device holds nSTATUS low for the time of the POR delay.

(4) After power-up, before and during configuration, CONF_DONE is low.

(5) Do not leave DCLK floating after configuration. You can drive it high or low, whichever is more convenient.

(6) DATA[7..0] are available as user I/O pins after configuration except for some exceptions on Stratix IV GT devices. The state of these pins depends on the dual-purpose pin settings.

(7) If needed, you can pause DCLK by holding it low. When DCLK restarts, the external host must provide data on the DATA[7..0] pins prior to sending the first DCLK rising edge.

Table 10–5 lists the timing parameters for Stratix IV devices for an FPP configuration when you enable the decompression and/or the design security features.

Table 10–5. FPP Timing Parameters for Stratix IV Devices with the Decompression and/or Design Security Features Enabled *(Note 1), (2)* (Part 1 of 2)

Symbol	Parameter	Minimum			Maximum			Units
		Stratix IV *(7)*	Stratix IV *(8)*	Stratix IV *(9)*	Stratix IV *(7)*	Stratix IV *(8)*	Stratix IV *(9)*	
t_{CF2CD}	nCONFIG low to CONF_DONE low	—			800			ns
t_{CF2ST0}	nCONFIG low to nSTATUS low	—			800			ns
t_{CFG}	nCONFIG low pulse width	2			—			µs
t_{STATUS}	nSTATUS low pulse width	10			500 *(3)*			µs
t_{CF2ST1}	nCONFIG high to nSTATUS high	—			500 *(4)*			µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	500			—			µs

Table 10–5. FPP Timing Parameters for Stratix IV Devices with the Decompression and/or Design Security Features Enabled *(Note 1)*, *(2)* (Part 2 of 2)

Symbol	Parameter	Minimum			Maximum			Units
		Stratix IV *(7)*	Stratix IV *(8)*	Stratix IV *(9)*	Stratix IV *(7)*	Stratix IV *(8)*	Stratix IV *(9)*	
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2			—			µs
t_{DSU}	Data setup time before rising edge on DCLK	4			—			ns
t_{DH}	Data hold time after rising edge on DCLK	3/(DCLK frequency) + 1			—			s
t_{DATA}	Data rate	—			250			Mbps
t_R	Input rise time	—			40			ns
t	Input fall time	—			40			ns
t_{CD2UM}	CONF_DONE high to user mode *(5)*	55			150			µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period			—			—
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on *(5)*	t_{CD2CU} + (8532 × CLKUSR period)			—			—
t_{CH}	DCLK high time *(6)*	3.6	4.5	5.6	—			ns
t_{CL}	DCLK low time *(6)*	3.6	4.5	5.6	—			ns
t_{CLK}	DCLK period *(6)*	8	10	12.5	—			ns
f_{MAX}	DCLK frequency	—			125	100	80	MHz

Notes to Table 10–5:

(1) This information is preliminary.
(2) Use these timing parameters when you use the decompression and/or design security features.
(3) You can obtain this value if you do not delay the configuration by extending the nCONFIG or nSTATUS low pulse width.
(4) This value is applicable if you do not delay the configuration by externally holding nSTATUS low.
(5) The minimum and maximum numbers apply only if you chose the internal oscillator as the clock source for starting the device.
(6) Adding up t_{CH} and t_{CL} equals to t_{CLK}. When EP4SE230 t_{CH} is 3.6 ns (min), t_{CL} must be 4.4 ns and vice versa.
(7) Applicable for EP4SE230, EP4SE360, EP4SGX70, EP4SGX110, EP4SGX180, EP4SGX230, EP4SGX290 (except F45 package), EP4SGX360 (except F45 package), EP4S40G2, EP4S100G2 devices.
(8) Applicable for EP4SE530, EP4SGX290 (only for F45 package), EP4SGX360 (only for F45 package), EP4SGX530, EP4S40G5, EP4S100G3, EP4S100G4, EP4S100G5 devices.
(9) Applicable to EP4SE820 only.

For more information about device configuration options and how to create configuration files, refer to the *Device Configuration Options and Configuration File Formats* chapters in volume 2 of the *Configuration Handbook*.

FPP Configuration Using a Microprocessor

In this configuration scheme, a microprocessor can control the transfer of configuration data from a storage device, such as flash memory, to the target Stratix IV device.

All information in "FPP Configuration Using a MAX II Device as an External Host" on page 10–6 is also applicable when using a microprocessor as an external host. Refer to this section for all configuration and timing information.

Fast Active Serial Configuration (Serial Configuration Devices)

In the fast AS configuration scheme, Stratix IV devices are configured using a serial configuration device. These configuration devices are low-cost devices with non-volatile memory that feature a simple four-pin interface and a small form factor.

The largest serial configuration device currently supports 128 MBits of configuration bitstream. Use the Stratix IV decompression features or select an FPP or PS configuration scheme for EP4SE360, EP4SGX290, EP4S40G5, EP4S100G3 and larger devices.

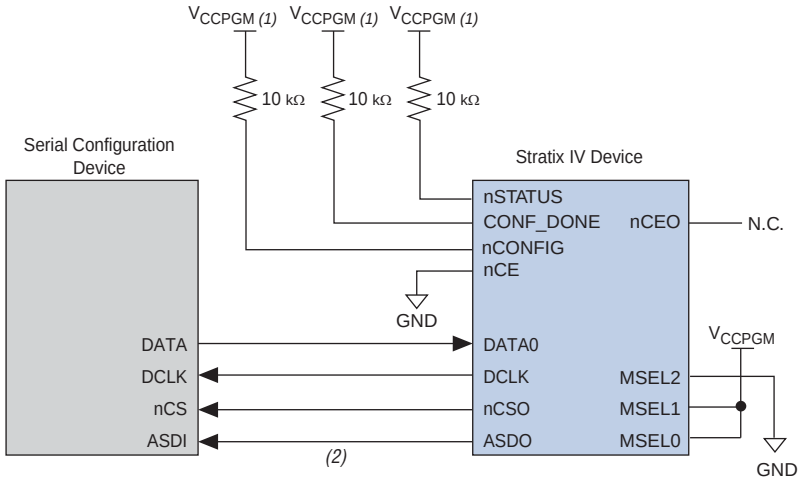
For more information about serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

Serial configuration devices provide a serial interface to access configuration data. During device configuration, Stratix IV devices read configuration data using the serial interface, decompress data if necessary, and configure their SRAM cells. This scheme is referred to as the AS configuration scheme because the Stratix IV device controls the configuration interface. This scheme contrasts with the PS configuration scheme where the configuration device controls the interface.

The Stratix IV decompression and design security features are fully available when configuring your Stratix IV device using fast AS mode.

Serial configuration devices have a four-pin interface—serial clock input (DCLK), serial data output (DATA), AS data input (ASDI), and an active-low chip select (nCS). This four-pin interface connects to Stratix IV device pins, as shown in Figure 10–6.

Figure 10–6. Single Device Fast AS Configuration



Notes to Figure 10–6:

(1) Connect the pull-up resistors to V_{CCPGM} at a 3.0-V supply.

(2) Stratix IV devices use the ASDO-to-ASDI path to control the configuration device.

You can power the EPCS serial configuration device with 3.0 V when you configure the Stratix IV FPGA using Active Serial (AS) configuration mode. This is feasible because the power supply to the EPCS device ranges between 2.7 V and 3.6 V. You do not need a dedicated 3.3 V power supply to power the EPCS device. The EPCS device and the VCCPGM pins on the Stratix IV device may share the same 3.0 V power supply.

After power-up, the Stratix IV devices go through a POR. The POR delay depends on the PORSEL pin setting. When PORSEL is driven low, the standard POR time is 100 ms < T_{POR} < 300 ms. When PORSEL is driven high, the fast POR time is 4 ms < T_{POR} < 12 ms. During POR, the device resets, holds nSTATUS and CONF_DONE low, and tri-states all user I/O pins. After the device successfully exits POR, all the user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors, which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

The configuration cycle consists of three stages—reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. After POR, the Stratix IV device releases nSTATUS, which is pulled high by an external 10-kΩ pull-up resistor and enters configuration mode.

To begin configuration, power the V_{CC}, V_{CCIO}, V_{CCPGM}, and V_{CCPD} voltages (for the banks where the configuration pins reside) to the appropriate voltage levels.

The serial clock (DCLK) generated by the Stratix IV device controls the entire configuration cycle and provides timing for the serial interface. Stratix IV devices use an internal oscillator to generate DCLK. Using the MSEL[] pins, you can select to use a 40 MHz oscillator.

In fast AS configuration schemes, Stratix IV devices drive out control signals on the falling edge of DCLK. The serial configuration device responds to the instructions by driving out configuration data on the falling edge of DCLK. Then the data is latched into the Stratix IV device on the following falling edge of DCLK.

In configuration mode, Stratix IV devices enable the serial configuration device by driving the nCSO output pin low, which connects to the chip select (nCS) pin of the configuration device. The Stratix IV device uses the serial clock (DCLK) and serial data output (ASDO) pins to send operation commands and/or read address signals to the serial configuration device. The configuration device provides data on its serial data output (DATA) pin, which connects to the DATA0 input of the Stratix IV devices.

After all the configuration bits are received by the Stratix IV device, it releases the open-drain CONF_DONE pin, which is pulled high by an external 10-kΩ resistor. Initialization begins only after the CONF_DONE signal reaches a logic high level. All AS configuration pins (DATA0, DCLK, nCSO, and ASDO) have weak internal pull-up resistors that are always active. After configuration, these pins are set as input tri-stated and are driven high by the weak internal pull-up resistors. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

In Stratix IV devices, the initialization clock source is either the internal oscillator or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If you use the internal oscillator, the Stratix IV device provides itself with enough clock cycles for proper initialization. You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. You can turn on the **Enable user-supplied start-up clock** (**CLKUSR**) option in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box. When you select the **Enable user supplied start-up clock** option, the CLKUSR pin is the initialization clock source. Supplying a clock on CLKUSR does not affect the configuration process. After all configuration data is accepted and CONF_DONE goes high, CLKUSR is enabled after four clock cycles of DCLK. After this time period elapses, Stratix IV devices require 8,532 clock cycles to initialize properly and enter user mode. Stratix IV devices support a CLKUSR f_{MAX} of 125 MHz.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box. If you use the INIT_DONE pin, it is high due to an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. After the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. This low-to-high transition signals that the device has entered user mode. When initialization is complete, the device enters user mode. In user mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

If an error occurs during configuration, Stratix IV devices assert the nSTATUS signal low, indicating a data frame error, and the CONF_DONE signal stays low. If the **Auto-restart configuration after error** option (available in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box) is turned on, the Stratix IV device resets the configuration device by pulsing nCSO, releases nSTATUS after a reset time-out period (a maximum of 500 µs), and retries configuration. If this option is turned off, the system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 2 µs to restart configuration.

If you have enabled the **Auto-restart configuration after error** option, the nSTATUS pin transitions from high to low and back again to high when a configuration error is detected. This appears as a low pulse at the nSTATUS pin with a minimum pulse width of 10 µs to a maximum pulse width of 500 µs, as defined in the t_{STATUS} specification.

When the Stratix IV device is in user mode, you can initiate reconfiguration by pulling the nCONFIG pin low. The nCONFIG pin must be low for at least 2 µs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. After nCONFIG returns to a logic high level and nSTATUS is released by the Stratix IV device, reconfiguration begins.

If you wish to gain control of the EPCS pins, hold the nCONFIG pin low and pull the nCE pin high. This causes the device to reset and tri-state the AS configuration pins.

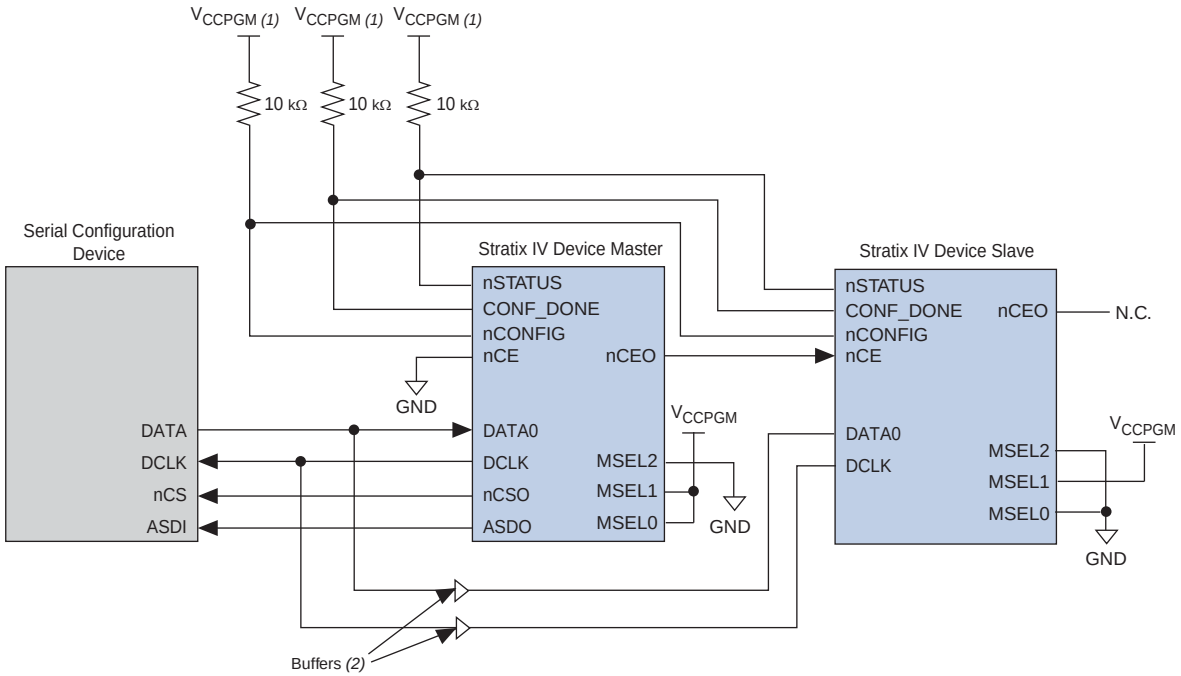
The timing parameters for AS mode are not listed here because the t_{CF2CD}, t_{CF2ST0}, t_{CFG}, t_{STATUS}, t_{CF2ST1}, and t_{CD2UM} timing parameters are identical to the timing parameters for PS mode listed in Table 10–7 on page 10–30.

You can configure multiple Stratix IV devices using a single serial configuration device. You can cascade multiple Stratix IV devices using the chip-enable (nCE) and chip-enable-out (nCEO) pins. The first device in the chain must have its nCE pin connected to GND. You must connect its nCEO pin to the nCE pin of the next device in the chain. When the first device captures all of its configuration data from the bitstream, it drives the nCEO pin low, enabling the next device in the chain. You must leave the nCEO pin of the last device unconnected. The nCONFIG, nSTATUS, CONF_DONE, DCLK, and DATA0 pins of each device in the chain are connected (refer to Figure 10–7).

The first Stratix IV device in the chain is the configuration master and controls configuration of the entire chain. You must connect its MSEL pins to select the AS configuration scheme. The remaining Stratix IV devices are configuration slaves. You must connect their MSEL pins to select the PS configuration scheme. Any other Altera device that supports PS configuration can also be part of the chain as a configuration slave.

Figure 10–7 shows the pin connections for the multi-device fast AS configuration.

Figure 10–7. Multi-Device Fast AS Configuration



Notes to Figure 10–7:

(1) Connect the pull-up resistors to V_{CCPGM} at a 3.0-V supply.

(2) Connect the repeater buffers between the Stratix IV master and slave device(s) for DATA[0] and DCLK. This is to prevent potential signal integrity and clock skew problems.

As shown in Figure 10–7, the nSTATUS and CONF_DONE pins on all target devices are connected together with external pull-up resistors. These pins are open-drain bidirectional pins on the devices. When the first device asserts nCEO (after receiving all of its configuration data), it releases its CONF_DONE pin. But the subsequent devices in the chain keep this shared CONF_DONE line low until they have received their configuration data. When all target devices in the chain have received their configuration data and have released CONF_DONE, the pull-up resistor drives a high level on this line and all devices simultaneously enter initialization mode.

If an error occurs at any point during configuration, the nSTATUS line is driven low by the failing device. If you enable the **Auto-restart configuration after error** option, reconfiguration of the entire chain begins after a reset time-out period (a maximum of 500 µs). If you did not enable the **Auto-restart configuration after error** option, the external system must monitor nSTATUS for errors and then pulse nCONFIG low to restart configuration. The external system can pulse nCONFIG if it is under system control rather than tied to V_{CCGPM}.

If you have enabled the **Auto-restart configuration after error** option, the nSTATUS pin transitions from high to low and back again to high when a configuration error is detected. This appears as a low pulse at the nSTATUS pin with a minimum pulse width of 10 µs to a maximum pulse width of 500 µs, as defined in the t_{STATUS} specification.

While you can cascade Stratix IV devices, you cannot cascade or chain together serial configuration devices.

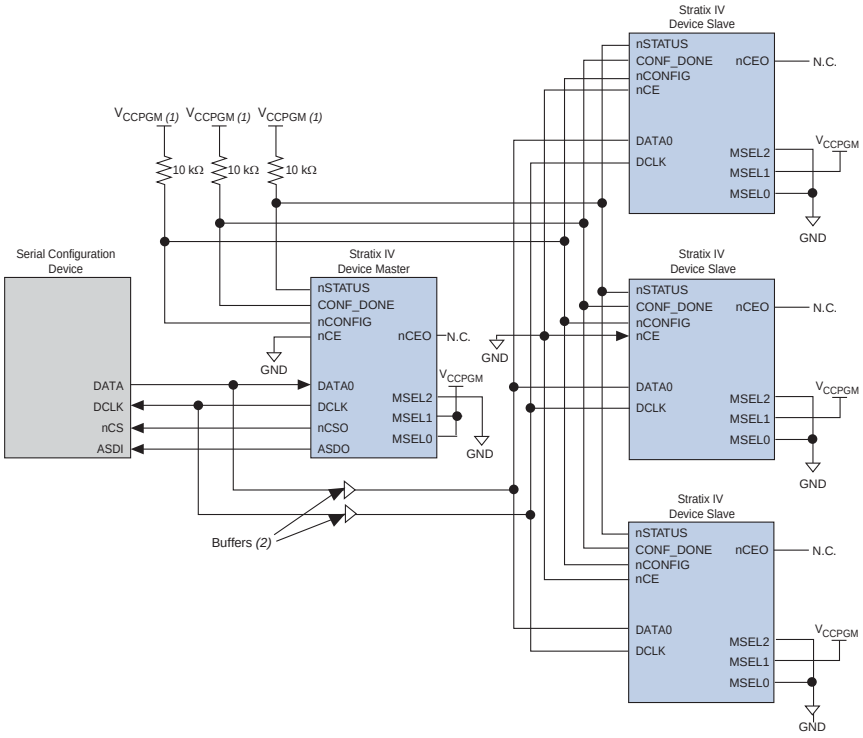
If the configuration bitstream size exceeds the capacity of a serial configuration device, you must select a larger configuration device and/or enable the compression feature. When configuring multiple devices, the size of the bitstream is the sum of the individual device configuration bitstreams.

A system may have multiple devices that contain the same configuration data. In active serial chains, you can implement this by storing one copy of the **.sof** in the serial configuration device. The same copy of the **.sof** configures the master Stratix IV device and all remaining slave devices concurrently. All Stratix IV devices must be the same density and package.

To configure four identical Stratix IV devices with the same **.sof**, set up the chain as shown in Figure 10–8. The first device is the master device and its MSEL pins must be set to select AS configuration. The other three slave devices are set up for concurrent configuration and their MSEL pins must be set to select PS configuration. The nCE input pins from the master and slave are connected to GND, and the DATA and DCLK pins connect in parallel to all four devices. During the configuration cycle, the master device reads its configuration data from the serial configuration device and transmits the configuration data to all three slave devices, configuring all of them simultaneously.

Figure 10–8 shows the multi-device fast AS configuration when the devices receive the same data using a single **.sof**.

Figure 10–8. Multi-Device Fast AS Configuration When the Devices Receive the Same Data Using a Single .sof



Notes to Figure 10–8:

(1) Connect the pull-up resistors to V_{CCPGM} at a 3.0-V supply.

(2) Connect the repeater buffers between the Stratix IV master and slave device(s) for `DATA[0]` and `DCLK`. This is to prevent potential signal integrity and clock skew problems.

Estimating Active Serial Configuration Time

Active serial configuration time is dominated by the time it takes to transfer data from the serial configuration device to the Stratix IV device. This serial interface is clocked by the Stratix IV `DCLK` output (generated from an internal oscillator) and must be set to **40 MHz (25 ns)**.Therefore, the minimum configuration time estimate for an EP4SE230 device (94, 600, 000 bits of uncompressed data) is:

RBF Size × (minimum `DCLK` period / 1 bit per `DCLK` cycle) = estimated minimum configuration time

94, 600, 000 bits × (25 ns / 1 bit) = 2365 ms

The calculation above is based on a preliminary uncompressed **.rbf** size. The final **.rbf** size will be available after the Quartus II software is able to generate the **.rbf**.

Enabling compression reduces the amount of configuration data that is transmitted to the Stratix IV device, which also reduces configuration time. On average, compression reduces configuration time, depending on the design.

Programming Serial Configuration Devices

Serial configuration devices are non-volatile, flash-memory-based devices. You can program these devices in-system using the USB-Blaster™, EthernetBlaster™, or ByteBlaster™ II download cable. Alternatively, you can program them using the Altera programming unit (APU), supported third-party programmers, or a microprocessor with the SRunner software driver.

You can perform in-system programming of serial configuration devices using the conventional AS programming interface or the JTAG interface solution.

Because serial configuration devices do not support the JTAG interface, the conventional method to program them is using the AS programming interface. The configuration data used to program serial configuration devices is downloaded using programming hardware.

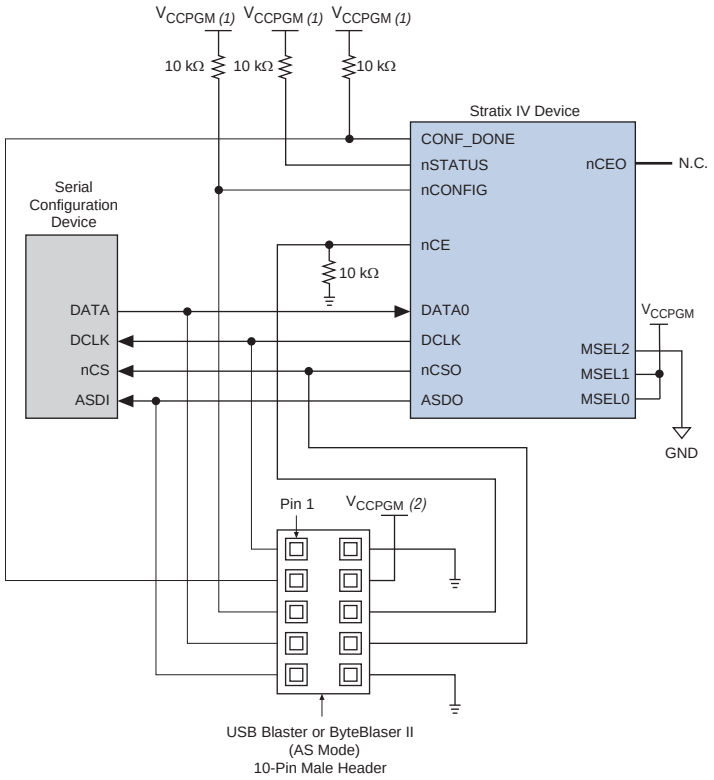
During in-system programming, the download cable disables device access to the AS interface by driving the nCE pin high. Stratix IV devices are also held in reset by a low level on nCONFIG. After programming is complete, the download cable releases nCE and nCONFIG, allowing the pull-down and pull-up resistors to drive GND and V_{CCPGM}, respectively. Figure 10–9 shows the download cable connections for the serial configuration device.

Altera has developed Serial FlashLoader (SFL), an in-system programming solution for serial configuration devices using the JTAG interface. This solution requires the Stratix IV device to be a bridge between the JTAG interface and the serial configuration device.

For more information about SFL, refer to *AN 370: Using the Serial FlashLoader with Quartus II Software*.

For more information about the USB-Blaster download cable, refer to the *USB-Blaster Download Cable User Guide*. For more information about the ByteBlaster II cable, refer to the *ByteBlaster II Download Cable User Guide*. For more information about the EthernetBlaster download cable, refer to the *EthernetBlaster Communications Cable User Guide.*

Figure 10–9. In-System Programming of Serial Configuration Devices



Notes to Figure 10–9:

(1) Connect these pull-up resistors to V_{CCPGM} at a 3.0-V supply.

(2) Power up the USB-ByteBlaster, ByteBlaster II, or EthernetBlaster cable's $V_{CC(TRGT)}$ with V_{CCPGM}.

You can program serial configuration devices with the Quartus II software using the Altera programming hardware and the appropriate configuration device programming adapter.

In production environments, you can program serial configuration devices using multiple methods. You can use Altera programming hardware or other third-party programming hardware to program blank serial configuration devices before they are mounted on PCBs. Alternatively, you can use an on-board microprocessor to program the serial configuration device in-system using C-based software drivers provided by Altera.

You can program a serial configuration device in-system by an external microprocessor using SRunner. SRunner is a software driver developed for embedded serial configuration device programming, which can be easily customized to fit in different embedded systems. SRunner is able to read raw programming data (.**rpd**) and write to serial configuration devices. The serial configuration device programming time using SRunner is comparable to the programming time with the Quartus II software.

For more information about SRunner, refer to *AN 418: SRunner: An Embedded Solution for Serial Configuration Device Programming* and the source code on the Altera website at www.altera.com.

For more information about programming serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

Guidelines for Connecting Serial Configuration Devices on an AS Interface

For single- and multi-device AS configurations, the board trace length and loading between the supported serial configuration device and the Stratix IV device family must follow the recommendations listed in Table 10–6.

Table 10–6. Maximum Trace Length and Loading for the AS Configuration

Stratix IV Device AS Pins	Maximum Board Trace Length from the Stratix IV Device to the Serial Configuration Device (Inches)	Maximum Board Load (pF)
DCLK	10	15
DATA[0]	10	30
nCSO	10	30
ASDO	10	30

Passive Serial Configuration

You can program a PS configuration for Stratix IV devices using an intelligent host, such as a MAX II device or microprocessor with flash memory, or a download cable. In the PS scheme, an external host (a MAX II device, embedded processor, or host PC) controls configuration. Configuration data is clocked into the target Stratix IV device using the DATA0 pin at each rising edge of DCLK.

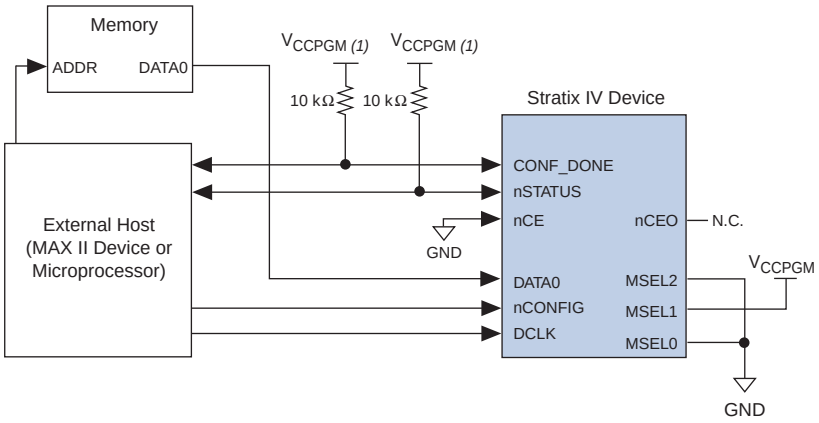
The Stratix IV decompression and design security features are fully available when configuring your Stratix IV device using PS mode.

PS Configuration Using a MAX II Device as an External Host

In this configuration scheme, you can use a MAX II device as an intelligent host that controls the transfer of configuration data from a storage device, such as flash memory, to the target Stratix IV device. You can store configuration data in **.rbf**, **.hex**, or **.ttf** format.

Figure 10–10 shows the configuration interface connections between a Stratix IV device and a MAX II device for single device configuration.

Figure 10–10. Single Device PS Configuration Using an External Host



Note to Figure 10–10:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Stratix IV device. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends powering up all configuration system I/Os with V_{CCPGM}.

After power-up, Stratix IV devices go through a POR. The POR delay depends on the PORSEL pin setting. When PORSEL is driven low, the standard POR time is 100 ms < T_{POR} < 300 ms. When PORSEL is driven high, the fast POR time is 4 ms < T_{POR} < 12 ms. During POR, the device resets, holds nSTATUS low, and tri-states all user I/O pins. After the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors that are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

The configuration cycle consists of three stages—reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration, the MAX II device must generate a low-to-high transition on the nCONFIG pin.

V_{CC}, V_{CCIO}, V_{CCPGM}, and V_{CCPD} of the banks where the configuration pins reside must be fully powered to the appropriate voltage levels to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. After nSTATUS is released, the device is ready to receive configuration data and the configuration stage begins. When nSTATUS is pulled high, the MAX II device places the configuration data one bit at a time on the DATA0 pin. If you are using configuration data in **.rbf**, **.hex**, or **.ttf** format, you must send the LSB of each data byte first. For example, if the **.rbf** contains the byte sequence 02 1B EE 01 FA, the serial bitstream you must transmit to the device is 0100-0000 1101-1000 0111-0111 1000-0000 0101-1111.

The Stratix IV device receives configuration data on the DATA0 pin and the clock is received on the DCLK pin. Data is latched into the device on the rising edge of DCLK. Data is continuously clocked into the target device until CONF_DONE goes high. After the device has received all configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by an external 10-kΩ pull-up resistor. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin. The CONF_DONE pin must have an external 10-kΩ pull-up resistor for the device to initialize.

In Stratix IV devices, the initialization clock source is either the internal oscillator or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If you use the internal oscillator, the Stratix IV device provides itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device. Driving DCLK to the device after configuration is complete does not affect device operation.

You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. You can turn on the **Enable user-supplied start-up clock** (**CLKUSR**) option in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box. If you supply a clock on CLKUSR, it will not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, CLKUSR is enabled after the time specified at t_{CD2CU}. After this time period elapses, Stratix IV devices require 8,532 clock cycles to initialize properly and enter user mode. Stratix IV devices support a CLKUSR f_{MAX} of 125 MHz.

An optional INIT_DONE pin is available that signals the end of initialization and the start of user-mode with a low-to-high transition. The **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box. If you use the INIT_DONE pin, it is high due to an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. After the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. The MAX II device must be able to detect this low-to-high transition that signals the device has entered user mode. When initialization is complete, the device enters user mode. In user-mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

Two DCLK falling edges are required after CONF_DONE goes high to begin the initialization of the device for both uncompressed and compressed bitstream in PS.

To ensure DCLK and DATA0 are not left floating at the end of configuration, the MAX II device must drive them either high or low, whichever is convenient on your board. The DATA[0] pin is available as a user I/O pin after configuration. When you chose the PS scheme as a default in the Quartus II software, this I/O pin is tri-stated in user mode and must be driven by the MAX II device. To change this default option in the Quartus II software, select the **Dual-Purpose Pins** tab of the **Device and Pin Options** dialog box.

The configuration clock (DCLK) speed must be below the specified frequency to ensure correct configuration. No maximum DCLK period exists, which means you can pause the configuration by halting DCLK for an indefinite amount of time.

If an error occurs during configuration, the device drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the MAX II device that there is an error. If the **Auto-restart configuration after error** option (available in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box) is turned on, the Stratix IV device releases nSTATUS after a reset time-out period (a maximum of 500 µs). After nSTATUS is released and pulled high by a pull-up resistor, the MAX II device can try to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

If you have enabled the **Auto-restart configuration after error** option, the nSTATUS pin transitions from high to low and back again to high when a configuration error is detected. This appears as a low pulse at the nSTATUS pin with a minimum pulse width of 10 µs to a maximum pulse width of 500 µs, as defined in the t_{STATUS} specification.

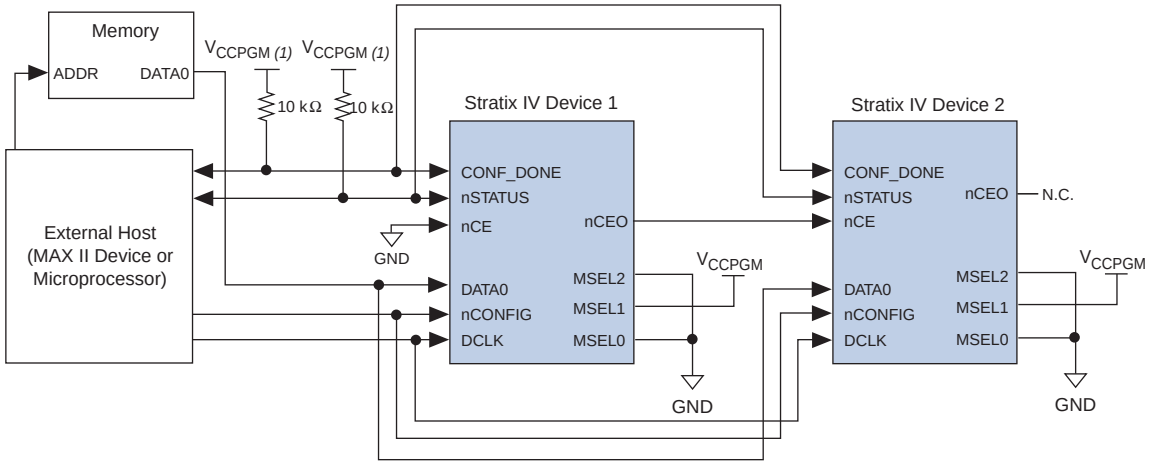
The MAX II device can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The CONF_DONE pin must be monitored by the MAX II device to detect errors and determine when programming completes. If all configuration data is sent, but CONF_DONE or INIT_DONE have not gone high, the MAX II device must reconfigure the target device.

If you use the optional CLKUSR pin and nCONFIG is pulled low to restart configuration during device initialization, you must ensure that CLKUSR continues toggling during the time nSTATUS is low (a maximum of 500 µs).

When the device is in user-mode, you can initiate a reconfiguration by transitioning the nCONFIG pin low-to-high. The nCONFIG pin must be low for at least 2 µs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. After nCONFIG returns to a logic high level and nSTATUS is released by the device, reconfiguration begins.

Figure 10–11 shows how to configure multiple devices using a MAX II device. This circuit is similar to the PS configuration circuit for a single device, except the Stratix IV devices are cascaded for multi-device configuration.

Figure 10–11. Multi-Device PS Configuration Using an External Host



Note to Figure 10–11:

(1) Connect the resistor to a supply that provides an acceptable input signal for all Stratix IV devices in the chain. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends powering up all configuration system I/Os with V_{CCPGM}.

In multi-device PS configuration, the first device's nCE pin is connected to GND, while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration within one clock cycle. Therefore, the transfer of data destinations is transparent to the MAX II device. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Because all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and you must reconfigure the entire chain. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

If the **Auto-restart configuration after error** option is turned on, the devices release their nSTATUS pins after a reset time-out period (a maximum of 500 µs). After all nSTATUS pins are released and pulled high, the MAX II device can try to reconfigure the chain without needing to pulse nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

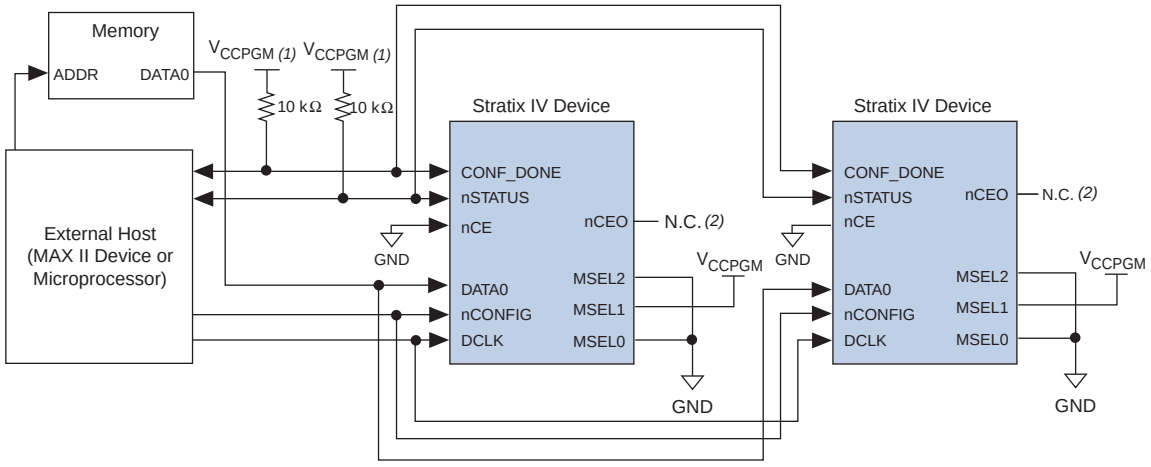
If you have enabled the **Auto-restart configuration after error** option, the nSTATUS pin transitions from high to low and back again to high when a configuration error is detected. This appears as a low pulse at the nSTATUS pin with a minimum pulse width of 10 µs to a maximum pulse width of 500 µs, as defined in the t_{STATUS} specification.

In your system, you can have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while the nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain.

Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices start and complete configuration at the same time.

Figure 10–12 shows multi-device PS configuration when both Stratix IV devices are receiving the same configuration data.

Figure 10–12. Multiple-Device PS Configuration When Both Devices Receive the Same Data



Notes to Figure 10–12:

(1) Connect the resistor to a supply that provides an acceptable input signal for all Stratix IV devices in the chain. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends powering up all configuration system I/Os with V_{CCPGM}.

(2) The nCEO pins of both devices are left unconnected when configuring the same configuration data into multiple devices.

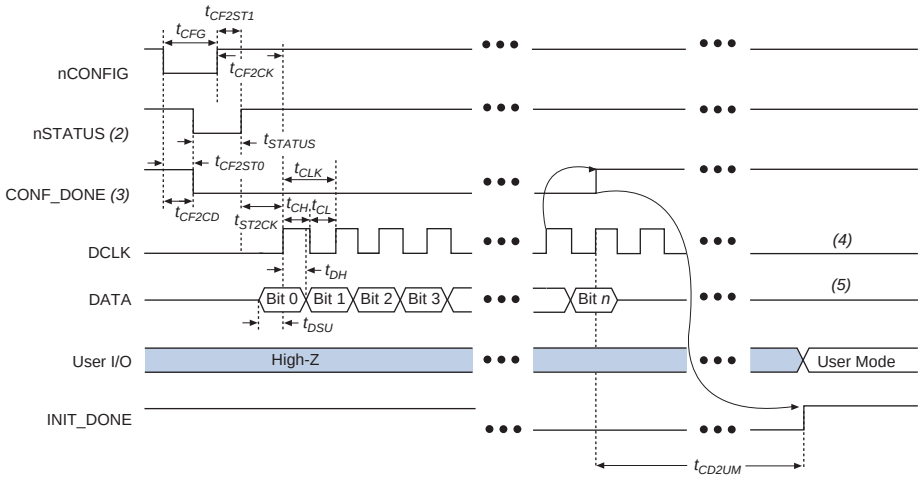
You can use a single configuration chain to configure Stratix IV devices with other Altera devices. To ensure that all devices in the chain complete configuration at the same time, or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information about configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook*.

PS Configuration Timing

Figure 10–13 shows the timing waveform for PS configuration when using a MAX II device as an external host.

Figure 10–13. PS Configuration Timing Waveform ***(Note 1)***



Notes to Figure 10–13:

(1) The beginning of this waveform shows the device in user mode. In user mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.

(2) After power-up, the Stratix IV device holds nSTATUS low for the time of the POR delay.

(3) After power-up, before and during configuration, CONF_DONE is low.

(4) Do not leave DCLK floating after configuration. You can drive it high or low, whichever is more convenient.

(5) DATA[0] is available as a user I/O pin after configuration. The state of this pin depends on the dual-purpose pin settings.

Table 10–7 lists the timing parameters for Stratix IV devices for PS configuration.

Table 10–7. PS Timing Parameters for Stratix IV Devices (Part 1 of 2) ***(Note 1)***

Symbol	Parameter	Minimum	Maximum	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low	—	800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low	—	800	ns
t_{CFG}	nCONFIG low pulse width	2	—	µs
t_{STATUS}	nSTATUS low pulse width	10	500 *(2)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high	—	500 *(3)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	500	—	µs
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2	—	µs
t_{DSU}	Data setup time before rising edge on DCLK	4	—	ns
t_{DH}	Data hold time after rising edge on DCLK	0	—	ns
t_{CH}	DCLK high time *(5)*	3.2	—	ns
t_{CL}	DCLK low time *(5)*	3.2	—	ns
t_{CLK}	DCLK period *(5)*	8	—	ns
f_{MAX}	DCLK frequency	—	125	MHz
t_R	Input rise time	—	40	ns

Table 10–7. PS Timing Parameters for Stratix IV Devices (Part 2 of 2) *(Note 1)*

Symbol	Parameter	Minimum	Maximum	Units
t_F	Input fall time	—	40	ns
t_{CD2UM}	CONF_DONE high to user mode *(4)*	55	150	s
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period	—	—
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	t_{CD2CU} + (8532 CLKUSR period)	—	—

Notes to Table 10–7:

(1) This information is preliminary.
(2) This value is applicable if you do not delay the configuration by extending the nCONFIG or nSTATUS low pulse width.
(3) This value is applicable if you do not delay the configuration by externally holding nSTATUS low.
(4) The minimum and maximum numbers apply only if you choose the internal oscillator as the clock source for starting the device.
(5) Adding up t_{CH} and t_{CL} equals to t_{CLK}. When t_{CH} is 3.2 ns (min), t_{CL} must be 4.8 ns and vice versa.

Device configuration options and how to create configuration files are described in the *Device Configuration Options and Configuration File Formats* chapters in volume 2 of the *Configuration Handbook*.

PS Configuration Using a Microprocessor

In this PS configuration scheme, a microprocessor controls the transfer of configuration data from a storage device, such as flash memory, to the target Stratix IV device.

For more information about configuration and timing information, refer to "PS Configuration Using a MAX II Device as an External Host" on page 10–25. This section is also applicable when using a microprocessor as an external host.

PS Configuration Using a Download Cable

In this section, the generic term "download cable" includes the Altera USB-Blaster universal serial bus (USB) port download cable, MasterBlaster serial/USB communications cable, ByteBlaster II parallel port download cable, ByteBlasterMV parallel port download cable, and EthernetBlaster download cable.

In a PS configuration with a download cable, an intelligent host (such as a PC) transfers data from a storage device to the device using the USB Blaster, MasterBlaster, ByteBlaster II, EthernetBlaster, or ByteBlasterMV cable.

After power-up, Stratix IV devices go through a POR. The POR delay depends on the PORSEL pin setting. When PORSEL is driven low, the standard POR time is 100 ms < T_{POR} < 300 ms. When PORSEL is driven high, the fast POR time is 4 ms < T_{POR} < 12 ms. During POR, the device resets, holds nSTATUS low, and tri-states all user I/O pins. After the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors, which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

The configuration cycle consists of three stages—reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration in this scheme, the download cable generates a low-to-high transition on the nCONFIG pin.

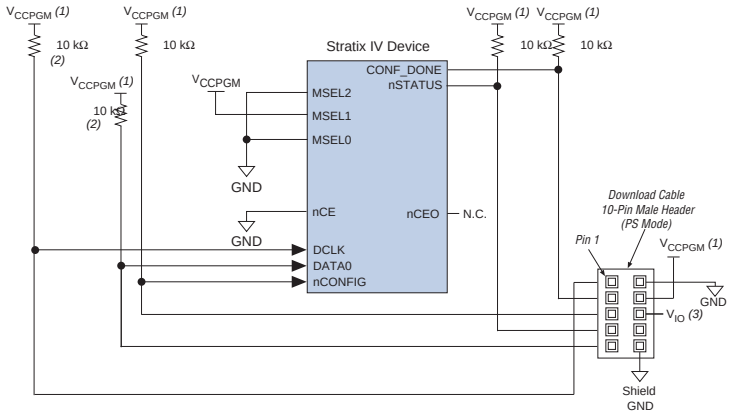
To begin configuration, power the V_{CC}, V_{CCIO}, V_{CCPGM}, and V_{CCPD} voltages (for the banks where the configuration pins reside) to the appropriate voltage levels.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. After nSTATUS is released, the device is ready to receive configuration data and the configuration stage begins. The programming hardware or download cable then places the configuration data one bit at a time on the device's DATA0 pin. The configuration data is clocked into the target device until CONF_DONE goes high. The CONF_DONE pin must have an external 10-kΩ pull-up resistor for the device to initialize.

When using a download cable, setting the **Auto-restart configuration after error** option does not affect the configuration cycle because you must manually restart configuration in the Quartus II software when an error occurs. Additionally, the **Enable user-supplied start-up clock** (**CLKUSR**) option has no affect on the device initialization because this option is disabled in the **.sof** when programming the device using the Quartus II programmer and download cable. Therefore, if you turn on the CLKUSR option, you do not need to provide a clock on CLKUSR when you are configuring the device with the Quartus II programmer and a download cable.

Figure 10–14 shows PS configuration for Stratix IV devices using a USB Blaster, EthernetBlaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV cable.

Figure 10–14. PS Configuration Using a USB Blaster, EthernetBlaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV Cable



Notes to Figure 10–14:

(1) Connect the pull-up resistor to the same supply voltage (V_{CCPGM}) as the USB Blaster, MasterBlaster (VIO pin), ByteBlaster II, ByteBlasterMV, or EthernetBlaster cable.

(2) You only need the pull-up resistors on DATA0 and DCLK if the download cable is the only configuration scheme used on your board. This ensures that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, you do not need the pull-up resistors on DATA0 and DCLK.

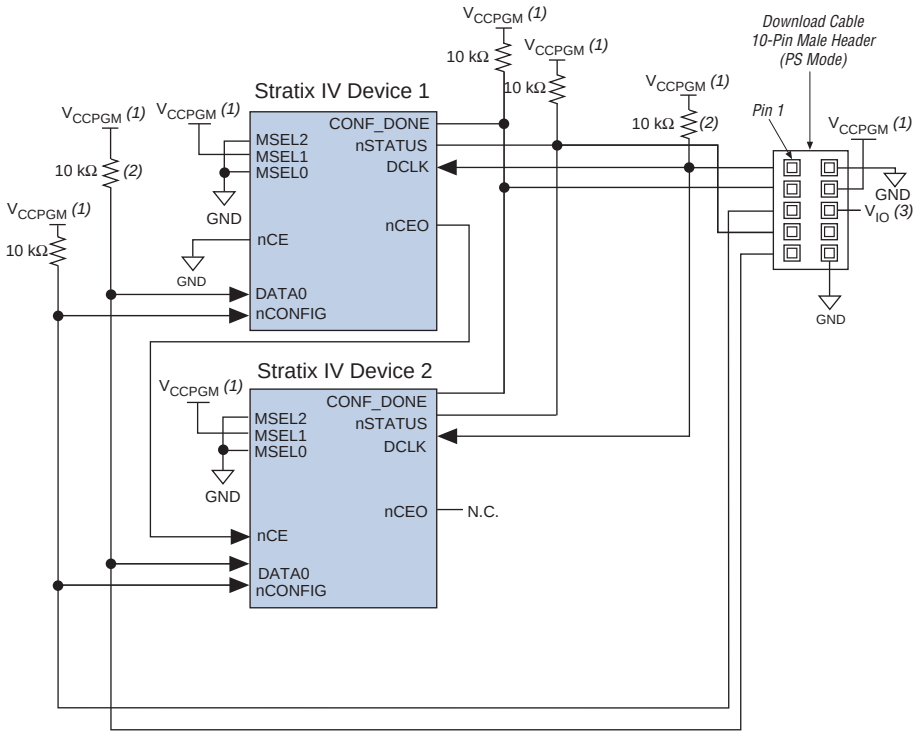
(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} must match the device's V_{CCPGM}. For more information about this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*. In the USB-Blaster, ByteBlaster II, and ByteBlasterMV cable, this pin is a no connect.

You can use a download cable to configure multiple Stratix IV devices by connecting each device's nCEO pin to the subsequent device's nCE pin. The first device's nCE pin is connected to GND, while its nCEO pin is connected to the nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. Because all CONF_DONE pins are tied together, all devices in the chain initialize and enter user mode at the same time.

In addition, because the nSTATUS pins are tied together, the entire chain halts configuration if any device detects an error. The **Auto-restart configuration after error** option does not affect the configuration cycle because you must manually restart the configuration in the Quartus II software when an error occurs.

Figure 10–15 shows how to configure multiple Stratix IV devices with a download cable.

Figure 10–15. Multi-Device PS Configuration Using a USB Blaster, EthernetBlaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV Cable



Notes to Figure 10–15:

(1) Connect the pull-up resistor to the same supply voltage (V_{CCPGM}) as the USB Blaster, MasterBlaster (VIO pin), ByteBlaster II, ByteBlasterMV, or EthernetBlaster cable.

(2) You only need the pull-up resistors on DATA0 and DCLK if the download cable is the only configuration scheme used on your board. This is to ensure that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, you do not need the pull-up resistors on DATA0 and DCLK.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} must match the device's V_{CCPGM}. For more information about this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*. In the USB-Blaster, ByteBlaster II, and ByteBlasterMV cables, this pin is a no connect.

For more information about how to use the USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV cables, refer to the following user guides:

- USB-Blaster Download Cable User Guide
- MasterBlaster Serial/USB Communications Cable User Guide
- ByteBlaster II Download Cable User Guide
- ByteBlasterMV Download Cable User Guide
- EthernetBlaster Communications Cable User Guide

JTAG Configuration

JTAG has developed a specification for boundary-scan testing. This boundary-scan test (BST) architecture offers the capability to efficiently test components on PCBs with tight lead spacing. The BST architecture can test pin connections without using physical test probes and capture functional data while a device is operating normally. You can also use JTAG circuitry to shift configuration data into the device. The Quartus II software automatically generates **.sof**s that you can use for JTAG configuration with a download cable in the Quartus II software programmer.

For more information about JTAG boundary-scan testing and commands available using Stratix IV devices, refer to the following documents:

- *JTAG Boundary Scan Testing in Stratix IV Devices* chapter
- Programming Support for Jam STAPL Language

Stratix IV devices are designed such that JTAG instructions have precedence over any device configuration modes. Therefore, JTAG configuration can take place without waiting for other configuration modes to complete. For example, if you attempt JTAG configuration of Stratix IV devices during PS configuration, PS configuration is terminated and JTAG configuration begins.

You cannot use the Stratix IV decompression or design security features if you are configuring your Stratix IV device when using JTAG-based configuration.

A device operating in JTAG mode uses four required pins, `TDI`, `TDO`, `TMS`, and `TCK`, and one optional pin, `TRST`. The `TCK` pin has an internal weak pull-down resistor, while the `TDI`, `TMS`, and `TRST` pins have weak internal pull-up resistors (typically 25 kΩ). The JTAG output pin `TDO` and all JTAG input pins are powered by 2.5-V/3.0-V V_{CCPD}. All the JTAG pins only support the LVTTL I/O standard.

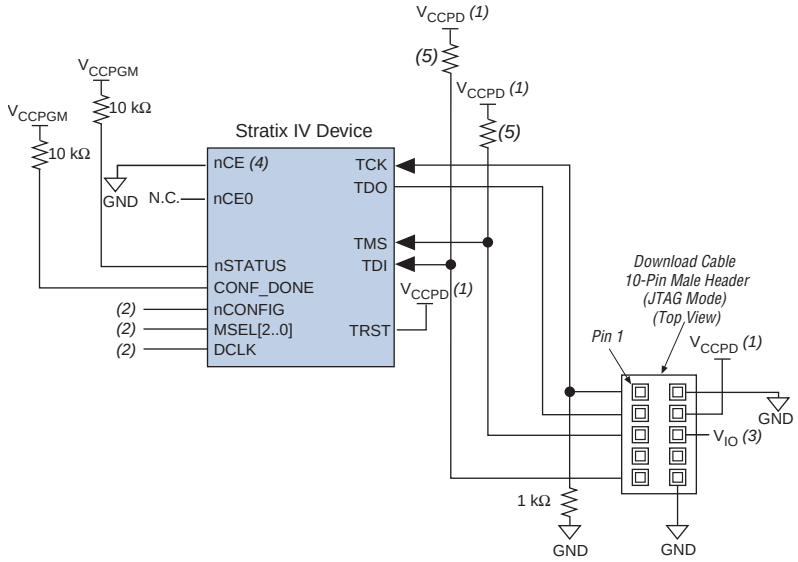
All user I/O pins are tri-stated during JTAG configuration.

All the JTAG pins are powered by the V_{CCPD} power supply of I/O bank 1A. For more information about how to connect a JTAG chain with multiple voltages across the devices in the chain, refer to the *JTAG Boundary Scan Testing in Stratix IV Devices* chapter.

During JTAG configuration, you can download data to the device on the PCB through the USB Blaster, MasterBlaster, ByteBlaster II, EthernetBlaster, or ByteBlasterMV download cable. Configuring devices through a cable is similar to programming devices in-system, except you must connect the TRST pin to V_{CCPD}. This ensures that the TAP controller is not reset.

Figure 10–16 shows JTAG configuration of a single Stratix IV device when using a download cable.

Figure 10–16. JTAG Configuration of a Single Device Using a Download Cable



Notes to Figure 10–16:

(1) Connect the pull-up resistor to the same supply voltage as the USB Blaster, MasterBlaster (VIO pin), ByteBlaster II, ByteBlasterMV, or EthernetBlaster cable. The voltage supply can be connected to the V_{CCPD} of the device.

(2) Connect the nCONFIG and MSEL[2..0] pins to support a non-JTAG configuration scheme. If you only use the JTAG configuration, connect nCONFIG to V_{CCPGM} and MSEL[2..0] to GND. Pull DCLK either high or low, whichever is convenient on your board.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} must match the device's V_{CCPD}. For more information about this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*. In the USB-Blaster, ByteBlaster II, and ByteBlasterMV cable, this pin is a no connect.

(4) You must connect nCE to GND or driven low for successful JTAG configuration.

(5) The pull-up resistor value can vary from 1 k to 10 kΩ.

To configure a single device in a JTAG chain, the programming software places all other devices in bypass mode. In bypass mode, devices pass programming data from the TDI pin to the TDO pin through a single bypass register without being affected internally. This scheme enables the programming software to program or verify the target device. Configuration data driven into the device appears on the TDO pin one clock cycle later.

The Quartus II software verifies successful JTAG configuration upon completion. At the end of configuration, the software checks the state of CONF_DONE through the JTAG port. When the Quartus II software generates a JAM file (**.jam)** for a multi-device chain, it contains instructions so that all the devices in the chain are initialized at the same time. If CONF_DONE is not high, the Quartus II software indicates that configuration has failed. If CONF_DONE is high, the software indicates that configuration was successful. After the configuration bitstream is transmitted serially using the JTAG TDI port, the TCK port is clocked an additional 1,094 cycles to perform device initialization.

Stratix IV devices have dedicated JTAG pins that always function as JTAG pins. Not only can you perform JTAG testing on Stratix IV devices before and after, but also during configuration. While other device families do not support JTAG testing during configuration, Stratix IV devices support the bypass, ID code, and sample instructions during configuration without interrupting configuration. All other JTAG instructions may only be issued by first interrupting configuration and reprogramming the I/O pins using the CONFIG_IO instruction.

The CONFIG_IO instruction allows I/O buffers to be configured using the JTAG port and when issued, interrupts configuration. This instruction allows you to perform board-level testing prior to configuring the Stratix IV device or waiting for a configuration device to complete configuration. After configuration has been interrupted and JTAG testing is complete, you must reconfigure the part using JTAG (PULSE_CONFIG instruction) or by pulsing nCONFIG low.

The chip-wide reset (DEV_CLRn) and chip-wide output enable (DEV_OE) pins on Stratix IV devices do not affect JTAG boundary-scan or programming operations. Toggling these pins does not affect JTAG operations (other than the usual boundary-scan operation).

When designing a board for JTAG configuration for Stratix IV devices, consider the dedicated configuration pins. Table 10–8 lists how these pins are connected during JTAG configuration.

Table 10–8. Dedicated Configuration Pin Connections During JTAG Configuration (Part 1 of 2)

Signal	Description
nCE	On all Stratix IV devices in the chain, nCE must be driven low by connecting it to GND, pulling it low using a resistor, or driving it by some control circuitry. For devices that are also in multi-device FPP, AS, or PS configuration chains, the nCE pins must be connected to GND during JTAG configuration or JTAG must be configured in the same order as the configuration chain.
nCEO	On all Stratix IV devices in the chain, you can leave nCEO floating or connected to the nCE of the next device.
MSEL	Do not leave these pins floating. These pins support whichever non-JTAG configuration is used in production. If you only use JTAG configuration, tie these pins to GND.

Table 10–8. Dedicated Configuration Pin Connections During JTAG Configuration (Part 2 of 2)

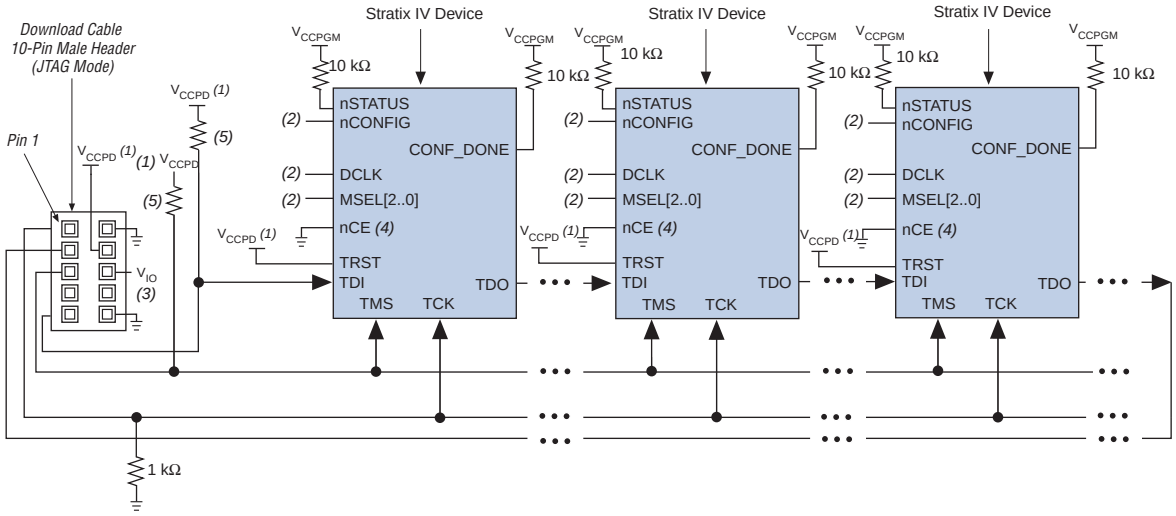
Signal	Description
nCONFIG	Driven high by connecting to V_{CCPGM}, pulling up using a resistor, or driven high by some control circuitry.
nSTATUS	Pull to V_{CCPGM} using a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each nSTATUS pin must be pulled up to V_{CCPGM} individually.
CONF_DONE	Pull to V_{CCPGM} using a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each CONF_DONE pin must be pulled up to V_{CCPGM} individually. CONF_DONE going high at the end of JTAG configuration indicates successful configuration.
DCLK	Do not leave DCLK floating. Drive low or high, whichever is more convenient on your board.

When programming a JTAG device chain, one JTAG-compatible header is connected to several devices. The number of devices in the JTAG chain is limited only by the drive capability of the download cable. When four or more devices are connected in a JTAG chain, Altera recommends buffering the TCK, TDI, and TMS pins with an on-board buffer.

JTAG-chain device programming is ideal when the system contains multiple devices, or when testing your system using JTAG BST circuitry.

Figure 10–17 shows a multi-device JTAG configuration when using a download cable.

Figure 10–17. JTAG Configuration of Multiple Devices Using a Download Cable



Notes to Figure 10–17:

(1) Connect the pull-up resistor to the same supply voltage as the USB Blaster, MasterBlaster (V_{IO} pin), ByteBlaster II, ByteBlasterMV, or EthernetBlaster cable. Connect the voltage supply to V_{CCPD} of the device.

(2) Connect the nCONFIG and MSEL[2..0] pins to support a non-JTAG configuration scheme. If you only use a JTAG configuration, connect nCONFIG to V_{CCPGM} and MSEL[2..0] to GND. Pull DCLK either high or low, whichever is convenient on your board.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} must match the device's V_{CCPD}. For more information about this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*. In the USB-Blaster, ByteBlaster II, and ByteBlasterMV cables, this pin is a no connect.

(4) You must connect nCE to GND or drive it low for successful JTAG configuration.

(5) The pull-up resistor value can vary from 1 k to 10 kΩ.

You must connect the nCE pin to GND or drive it low during JTAG configuration. In multi-device FPP, AS, and PS configuration chains, the first device's nCE pin is connected to GND, while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. In addition, the CONF_DONE and nSTATUS signals are all shared in multi-device FPP, AS, or PS configuration chains so the devices can enter user mode at the same time after configuration is complete. When the CONF_DONE and nSTATUS signals are shared among all the devices, you must configure every device when JTAG configuration is performed.

If you only use JTAG configuration, Altera recommends connecting the circuitry as shown in Figure 10–17, where each of the CONF_DONE and nSTATUS signals are isolated, so that each device can enter user mode individually.

After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. Therefore, if these devices are also in a JTAG chain, ensure the nCE pins are connected to GND during JTAG configuration or that the devices are JTAG configured in the same order as the configuration chain. As long as the devices are JTAG configured in the same order as the multi-device configuration chain, the nCEO of the previous device drives the nCE of the next device low when it has successfully been JTAG configured.

You can place other Altera devices that have JTAG support in the same JTAG chain for device programming and configuration.

JTAG configuration support is enhanced and allows more than 17 Stratix IV devices to be cascaded in a JTAG chain.

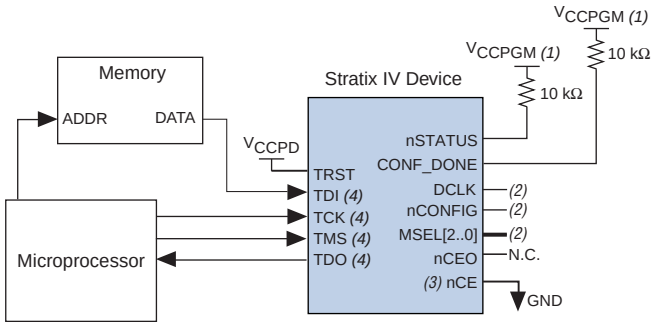
For more information about configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook*.

You can configure Stratix IV devices using multiple configuration schemes on the same board. Combining JTAG configuration with AS configuration on your board is useful in the prototyping environment because it allows multiple methods to configure your FPGA.

For more information about combining JTAG configuration with other configuration schemes, refer to the *Combining Different Configuration Schemes* chapter in volume 2 of the *Configuration Handbook*.

Figure 10–18 shows JTAG configuration of a Stratix IV device using a microprocessor.

Figure 10–18. JTAG Configuration of a Single Device Using a Microprocessor



Notes to Figure 10–18:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for all Stratix IV devices in the chain. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device.

(2) Connect the nCONFIG and MSEL[2..0] pins to support a non-JTAG configuration scheme. If you use only a JTAG configuration, connect nCONFIG to V_{CCGPM} and MSEL[2..0] to GND. Pull DCLK either high or low, whichever is convenient on your board.

(3) Connect nCE to GND or drive it low for successful JTAG configuration.

(4) The microprocessor must use the same I/O standard as V_{CCPD} to drive the JTAG pins.

Jam STAPL

Jam™ STAPL, JEDEC standard JESD-71, is a standard file format for in-system programmability (ISP) purposes. Jam STAPL supports programming or configuration of programmable devices and testing of electronic systems, using the IEEE 1149.1 JTAG interface. Jam STAPL is a freely licensed open standard.

The Jam Player provides an interface for manipulating the IEEE Std. 1149.1 JTAG TAP state machine.

For more information about JTAG and Jam STAPL in embedded environments, refer to *Using Jam STAPL for ISP via an Embedded Processor*. To download the Jam Player, visit the Altera website at www.altera.com.

Device Configuration Pins

The following tables list the connections and functionality of all the configuration-related pins on Stratix IV devices. Table 10–9 lists the Stratix IV configuration pins and their power supply.

Table 10–9. Stratix IV Configuration Pin Summary (Part 1 of 2) *(Note 1)*

Description	Input/Output	Dedicated	Powered By	Configuration Mode
TDI	Input	Yes	V_{CCPD}	JTAG
TMS	Input	Yes	V_{CCPD}	JTAG
TCK	Input	Yes	V_{CCPD}	JTAG
TRST	Input	Yes	V_{CCPD}	JTAG
TDO	Output	Yes	V_{CCPD}	JTAG
CRC_ERROR	Output	—	Pull-up	Optional, all modes

Table 10–9. Stratix IV Configuration Pin Summary (Part 2 of 2) ***(Note 1)***

Description	Input/Output	Dedicated	Powered By	Configuration Mode
DATA0	Input	—	V_{CCPGM}/V_{CCIO} *(3)*	All modes except JTAG
DATA[7..1]	Input	—	V_{CCPGM}/V_{CCIO} *(3)*	FPP
INIT_DONE	Output	—	Pull-up	Optional, all modes
CLKUSR	Input	—	V_{CCPGM}/V_{CCIO} *(3)*	Optional
nSTATUS	Bidirectional	Yes	V_{CCPGM}/Pull-up	All modes
nCE	Input	Yes	V_{CCPGM}	All modes
CONF_DONE	Bidirectional	Yes	V_{CCPGM}/Pull-up	All modes
nCONFIG	Input	Yes	V_{CCPGM}	All modes
PORSEL	Input	Yes	V_{CC} *(2)*	All modes
ASDO *(4)*	Output	Yes	V_{CCPGM}	AS
nCSO *(4)*	Output	Yes	V_{CCPGM}	AS
DCLK *(4)*	Input	Yes	V_{CCPGM}	PS, FPP
	Output	Yes	V_{CCPGM}	AS
nIO_PULLUP	Input	Yes	V_{CC} *(2)*	All modes
nCEO	Output	Yes	V_{CCPGM}	All modes
MSEL[2..0]	Input	Yes	V_{CC} *(2)*	All modes

Notes to Table 10–9:

(1) The total number of pins is 29. The total number of dedicated pins is 18.

(2) Although MSEL[2..0], PORSEL, and nIO_PULLUP are powered up by V_{CC}, Altera recommends connecting these pins to V_{CCPGM} or GND directly without using a pull-up or pull-down resistor.

(3) These pins are powered up by V_{CCPGM} during configuration. These pins are powered up by V_{CCIO} if they are used as regular I/O in user mode.

(4) To tri-state this pin, in the Quartus II software, on the Assignments menu, select **Device**. On the **Device** page, select **Device and Pin Options...** On the **Device and Pin Options** page, select **Configuration** and select the **Enable input tri-state on active configuration pins in user mode** option.

Table 10–10 lists the dedicated configuration pins. You must connect these pins properly on your board for successful configuration. Some of these pins may not be required for your configuration schemes.

Table 10–10. Dedicated Configuration Pins on the Stratix IV Device (Part 1 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
VCCPGM	N/A	All	Power	Dedicated power pin. Use this pin to power all dedicated configuration inputs, dedicated configuration outputs, dedicated configuration bidirectional pins, and some of the dual functional pins that are used for configuration. You must connect this pin to 1.8, 2.5, or 3.0 V. V_{CCPGM} must ramp-up from 0 V to V_{CCPGM} within 100 ms when PORSEL is low or 4 ms when PORSEL is high. If V_{CCPGM} is not ramped up within this specified time, your Stratix IV device will not configure successfully. If your system does not allow a V_{CCPGM} ramp-up within 100 ms or 4 ms, you must hold nCONFIG low until all power supplies are stable.
VCCPD	N/A	All	Power	Dedicated power pin. Use this pin to power the I/O pre-drivers, JTAG input and output pins, and design security circuitry. You must connect this pin to 2.5 V or 3.0 V, depending on the I/O standards selected. For the 3.0-V I/O standard, V_{CCPD} = 3.0 V. For the 2.5 V or below I/O standards, V_{CCPD} = 2.5 V. V_{CCPD} must ramp-up from 0 V to 2.5 V / 3.0 V within 100 ms when PORSEL is low or 4 ms when PORSEL is high. If V_{CCPD} is not ramped up within this specified time, your Stratix IV device will not configure successfully. If your system does not allow a V_{CCPD} to ramp-up time within 100 ms or 4 ms, you must hold nCONFIG low until all power supplies are stable.
PORSEL	N/A	All	Input	Dedicated input that selects between a standard POR time or a fast POR time. A logic low selects a standard POR time setting of 100 ms < T_{POR} < 300 ms and a logic high selects a fast POR time setting of 4 ms < T_{POR} < 12 ms. The PORSEL input buffer is powered by V_{CC} and has an internal 5-kΩ pull-down resistor that is always active. Tie the PORSEL pin directly to V_{CCPGM} or GND.
nIO_PULLUP	N/A	All	Input	Dedicated input that chooses whether the internal pull-up resistors on the user I/O pins and dual-purpose I/O pins (nCSO, nASDO, DATA[7..0], CLKUSR, and INIT_DONE) are on or off before and during configuration. A logic high turns off the weak internal pull-up resistors; a logic low turns them on. The nIO-PULLUP input buffer is powered by V_{CC} and has an internal 5-kΩ pull-down resistor that is always active. The nIO-PULLUP can be tied directly to V_{CCPGM}, using a 1-kΩ pull-up resistor or tied directly to GND, depending on your device requirements.

Table 10–10. Dedicated Configuration Pins on the Stratix IV Device (Part 2 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
MSEL[2..0]	N/A	All	Input	Three-bit configuration input that sets the Stratix IV device configuration scheme. For the appropriate connections, refer to Table 10–1 on page 10–2. You must hardwire these pins to V_{CCPGM} or GND. The MSEL[2..0] pins have internal 5-kΩ pull-down resistors that are always active.
nCONFIG	N/A	All	Input	Configuration control input. Pulling this pin low during user-mode causes the device to lose its configuration data, enter a reset state, and tri-state all I/O pins. Returning this pin to a logic high level initiates a reconfiguration. Configuration is possible only if this pin is high, except in JTAG programming mode, when nCONFIG is ignored.
nSTATUS	N/A	All	Bidirectional open-drain	The device drives nSTATUS low immediately after power-up and releases it after the POR time. During user mode and regular configuration, this pin is pulled high by an external 10-kΩresistor. This pin, when driven low by the Stratix IV device, indicates that the device has encountered an error during configuration. ■ Status output—If an error occurs during configuration, nSTATUS is pulled low by the target device. ■ Status input—If an external source drives the nSTATUS pin low during configuration or initialization, the target device enters an error state. Driving nSTATUS low after configuration and initialization does not affect the configured device. If you use a configuration device, driving nSTATUS low causes the configuration device to attempt to configure the device, but because the device ignores transitions on nSTATUS in user mode, the device does not reconfigure. To initiate a reconfiguration, nCONFIG must be pulled low. If you have enabled the **Auto-restart configuration after error** option, the nSTATUS pin transitions from high to low and back again to high when a configuration error is detected. This appears as a low pulse at the pin with a minimum pulse width of 10 µs to a maximum pulse width of 500 µs, as defined in the t_{STATUS} specification.

Table 10–10. Dedicated Configuration Pins on the Stratix IV Device (Part 3 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nSTATUS (continued)	—	—	—	If V_{CCPGM} is not fully powered up, the following could occur: ■ V_{CCPGM} is powered high enough for the nSTATUS buffer to function properly and nSTATUS is driven low. When V_{CCPGM} is ramped up, POR trips and nSTATUS is released after POR expires. ■ V_{CCPGM} is not powered high enough for the nSTATUS buffer to function properly. In this situation, nSTATUS might appear logic high, triggering a configuration attempt that would fail because POR did not yet trip. When V_{CCPD} is powered up, nSTATUS is pulled low because POR did not yet trip. When POR trips after V_{CCPGM} is powered up, nSTATUS is released and pulled high. At that point, reconfiguration is triggered and the device is configured.
CONF_DONE	N/A	All	Bidirectional open-drain	Status output. The target device drives the CONF_DONE pin low before and during configuration. After all the configuration data is received without error and the initialization cycle starts, the target device releases CONF_DONE. Status input. After all the data is received and CONF_DONE goes high, the target device initializes and enters user mode. The CONF_DONE pin must have an external 10-kΩ pull-up resistor for the device to initialize. Driving CONF_DONE low after configuration and initialization does not affect the configured device.
nCE	N/A	All	Input	Active-low chip enable. The nCE pin activates the device with a low signal to allow configuration. The nCE pin must be held low during configuration, initialization, and user mode. In single device configuration, it must be tied low. In multi-device configuration, nCE of the first device is tied low, while its nCEO pin is connected to nCE of the next device in the chain. The nCE pin must also be held low for successful JTAG programming of the device.
nCEO	N/A	All	Output	Output that drives low when device configuration is complete. In single device configuration, this pin is left floating. In multi-device configuration, this pin feeds the next device's nCE pin. The nCEO of the last device in the chain is left floating. The nCEO pin is powered by V_{CCPGM}.
ASDO	N/A	AS	Output	Control signal from the Stratix IV device to the serial configuration device in AS mode used to read out configuration data. In AS mode, ASDO has an internal pull-up resistor that is always active.

Table 10–10. Dedicated Configuration Pins on the Stratix IV Device (Part 4 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nCSO	N/A	AS	Output	Output control signal from the Stratix IV device to the serial configuration device in AS mode that enables the configuration device. In AS mode, nCSO has an internal pull-up resistor that is always active.
DCLK	N/A	Synchronous configuration schemes (PS, FPP, AS)	Input (PS, FPP) Output (AS)	In PS and FPP configurations, DCLK is the clock input used to clock data from an external source into the target device. Data is latched into the device on the rising edge of DCLK. In AS mode, DCLK is an output from the Stratix IV device that provides timing for the configuration interface. In AS mode, DCLK has an internal pull-up resistor (typically 25 kΩ) that is always active. In AS configuration schemes, this pin is driven into an inactive state after configuration completes. You can use this pin as a user I/O during user mode. In PS or FPP schemes that use a control host, you must drive DCLK either high or low, whichever is more convenient. In passive schemes, you cannot use DCLK as a user I/O during user mode. Toggling this pin after configuration does not affect the configured device.
DATA0	N/A in AS mode. I/O in PS or FPP mode.	PS, FPP, AS	Input	Data input. In serial configuration modes, bit-wide configuration data is presented to the target device on the DATA0 pin. In AS mode, DATA0 has an internal pull-up resistor that is always active. After PS or FPP configuration, DATA0 is available as a user I/O pin. The state of this pin depends on the **Dual-Purpose Pin** settings.
DATA[7..1]	I/O	Parallel configuration schemes (FPP)	Inputs	Data inputs. Byte-wide configuration data is presented to the target device on DATA[7..0]. In serial configuration schemes, they function as user I/O pins during configuration, which means they are tri-stated. After FPP configuration, DATA[7..1] are available as user I/O pins. The state of these pins depends on the **Dual-Purpose Pin** settings.

Table 10–11 lists the optional configuration pins. If these optional configuration pins are not enabled in the Quartus II software, they are available as general-purpose user I/O pins. Therefore, during configuration, these pins function as user I/O pins and are tri-stated with weak pull-up resistors.

Table 10–11. Optional Configuration Pins

Pin Name	User Mode	Pin Type	Description
CLKUSR	N/A if option is on. I/O if option is off.	Input	Optional user-supplied clock input synchronizes the initialization of one or more devices. Enable this pin by turning on the **Enable user-supplied start-up clock** (**CLKUSR**) option in the Quartus II software.
INIT_DONE	N/A if option is on. I/O if option is off.	Output open-drain	Use as a status pin to indicate when the device has initialized and is in user mode. When nCONFIG is low and during the beginning of configuration, the INIT_DONE pin is tri-stated and pulled high due to an external 10-kΩ pull-up resistor. After the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high and the device enters user mode. Thus, the monitoring circuitry must be able to detect a low-to-high transition. Enable this pin by turning on the **Enable INIT_DONE output** option in the Quartus II software.
DEV_OE	N/A if option is on. I/O if option is off.	Input	Optional pin that allows you to override all tri-states on the device. When this pin is driven low, all I/O pins are tri-stated. When this pin is driven high, all I/O pins behave as programmed. Enable this pin by turning on the **Enable device-wide output enable** (**DEV_OE**) option in the Quartus II software.
DEV_CLRn	N/A if option is on. I/O if option is off.	Input	Optional pin that allows you to override all clears on all device registers. When this pin is driven low, all registers are cleared. When this pin is driven high, all registers behave as programmed. Enable this pin by turning on the **Enable device-wide reset** (**DEV_CLRn**) option in the Quartus II software.

Table 10–12 lists the dedicated JTAG pins. JTAG pins must be kept stable before and during configuration to prevent accidental loading of JTAG instructions. The TDI, TMS, and TRST pins have weak internal pull-up resistors, while TCK has a weak internal pull-down resistor (typically 25 kΩ). If you plan to use the SignalTap® embedded logic array analyzer, you must connect the JTAG pins of the Stratix IV device to a JTAG header on your board.

Table 10–12. Dedicated JTAG Pins

Pin Name	User Mode	Pin Type	Description
TDI	N/A	Test data input	Serial input pin for instructions as well as test and programming data. Data is shifted on the rising edge of TCK. The TDI pin is powered by the 2.5-V/3.0-V V_{CCPD} supply. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by connecting this pin to logic high using a 1-kΩ resistor.
TDO	N/A	Test data output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. The TDO pin is powered by V_{CCPD}. For recommendations about connecting a JTAG chain with multiple voltages across the devices in the chain, refer to the *JTAG Boundary Scan Testing in Stratix IV Devices* chapter. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by leaving this pin unconnected.
TMS	N/A	Test mode select	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. TMS is evaluated on the rising edge of TCK. Therefore, you must set up TMS before the rising edge of TCK. Transitions within the state machine occur on the falling edge of TCK after the signal is applied to TMS. The TMS pin is powered by 2.5-V/3.0-V V_{CCPD}. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by connecting this pin to logic high using a 1-kΩ resistor.
TCK	N/A	Test clock input	Clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. The TCK pin is powered by the 2.5-V/3.0-V V_{CCPD} supply. It is expected that the clock input waveform have a nominal 50% duty cycle. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by connecting TCK to GND.
TRST	N/A	Test reset input (optional)	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to IEEE Std. 1149.1. The TRST pin is powered by the 2.5-V/3.0-V V_{CCPD} supply. Hold TMS at 1 or keep TCK static while TRST is changed from 0 to 1. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by connecting the TRST pin to GND.

For more information about the pin connection recommendations, refer to the *Stratix IV GX and Stratix IV E Device Family Pin Connection Guidelines*.

Configuration Data Decompression

Stratix IV devices support configuration data decompression, which saves configuration memory space and time. This feature allows you to store compressed configuration data in configuration devices or other memory and transmit this compressed bitstream to Stratix IV devices. During configuration, the Stratix IV device decompresses the bitstream in real time and programs its SRAM cells.

Preliminary data indicates that compression typically reduces the configuration bitstream size by 30% to 55% based on the designs used.

Stratix IV devices support decompression in the FPP (when using a MAX II device or microprocessor + flash), fast AS, and PS configuration schemes. The Stratix IV decompression feature is not available in the JTAG configuration scheme.

In PS mode, use the Stratix IV decompression feature because sending compressed configuration data reduces configuration time.

When you enable compression, the Quartus II software generates configuration files with compressed configuration data. This compressed file reduces the storage requirements in the configuration device or flash memory, and decreases the time needed to transmit the bitstream to the Stratix IV device. The time required by a Stratix IV device to decompress a configuration file is less than the time needed to transmit the configuration data to the device.

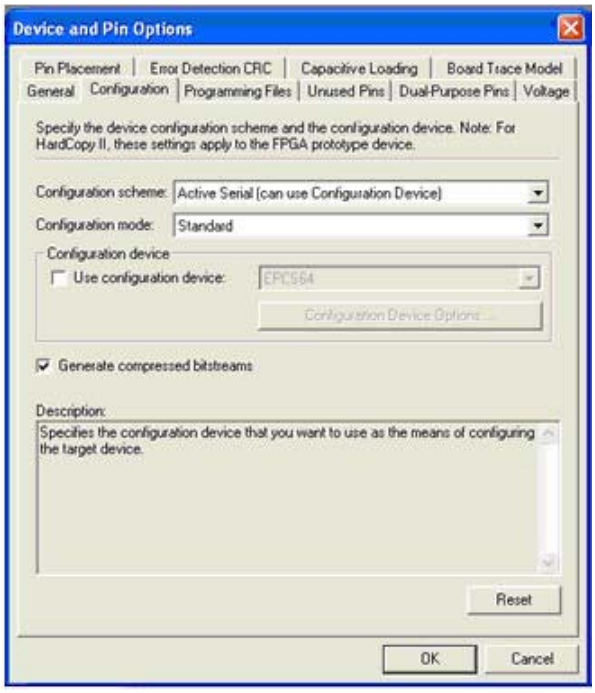
There are two ways to enable compression for Stratix IV bitstreams—before design compilation (in the Compiler Settings menu) and after design compilation (in the **Convert Programming Files** window).

To enable compression in the project's Compiler Settings menu, follow these steps:

1. On the Assignments menu, click **Device** to bring up the **Settings** dialog box.
2. After selecting your Stratix IV device, open the **Device and Pin Options** window.

3. In the **Configuration** settings tab, turn on **Generate compressed bitstreams** (as shown in Figure 10–19).

Figure 10–19. Enabling Compression for Stratix IV Bitstreams in Compiler Settings



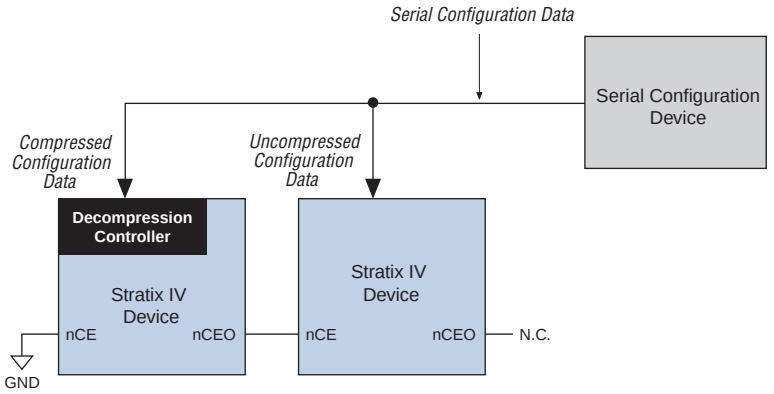
You can also enable compression when creating programming files from the **Convert Programming Files** window. To do this, follow these steps:

1. On the File menu, click **Convert Programming Files**.
2. Select the programming file type (**.pof**, **.sram**, **.hex**, **.rbf**, or **.ttf**).
3. For **.pof** output files, select a configuration device.
4. In the **Input files to convert** box, select **SOF Data**.
5. Select **Add File** and add a Stratix IV device **.sof** file.
6. Select the name of the file you added to the **SOF Data** area and click **Properties**.
7. Check the **Compression** check box.

When multiple Stratix IV devices are cascaded, you can selectively enable the compression feature for each device in the chain if you are using a serial configuration scheme. Figure 10–20 shows a chain of two Stratix IV devices. The first Stratix IV device has compression enabled; therefore, receives a compressed bitstream from the configuration device. The second Stratix IV device has the compression feature disabled and receives uncompressed data.

In a multi-device FPP configuration chain (with a MAX II device or microprocessor + flash), all Stratix IV devices in the chain must either enable or disable the decompression feature. You cannot selectively enable the compression feature for each device in the chain because of the DATA and DCLK relationship.

Figure 10–20. Compressed and Uncompressed Configuration Data in the Same Configuration File



You can generate programming files for this setup by clicking **Convert Programming Files** on the File menu in the Quartus II software.

Remote System Upgrades

This section describes the functionality and implementation of the dedicated remote system upgrade circuitry. It also defines several concepts related to remote system upgrade, including factory configuration, application configuration, remote update mode, and user watchdog timer. Additionally, this section provides design guidelines for implementing remote system upgrades with the supported configuration schemes.

System designers sometimes face challenges such as shortened design cycles, evolving standards, and system deployments in remote locations. Stratix IV devices help overcome these challenges with their inherent reprogrammability and dedicated circuitry to perform remote system upgrades. Remote system upgrades help deliver feature enhancements and bug fixes without costly recalls, reduce time-to-market, extend product life, and avoid system downtime.

Stratix IV devices feature dedicated remote system upgrade circuitry. Soft logic (either the Nios® II embedded processor or user logic) implemented in a Stratix IV device can download a new configuration image from a remote location, store it in configuration memory, and direct the dedicated remote system upgrade circuitry to initiate a reconfiguration cycle. The dedicated circuitry performs error detection during and after the configuration process, recovers from any error condition by reverting back to a safe configuration image, and provides error status information.

Remote system upgrade is supported in fast AS Stratix IV configuration schemes. You can also implement remote system upgrade in conjunction with advanced Stratix IV features such as real-time decompression of configuration data and design security using the advanced encryption standard (AES) for secure and efficient field upgrades. The largest serial configuration device currently supports 128 Mbits of configuration bitstream.

☞ Stratix IV devices only support remote system upgrade in the single device fast AS configuration scheme. Because the largest serial configuration device currently supports 128 Mbits of configuration bitstream, the remote system upgrade feature is not supported in EP4SGX290, EP4SE360, and larger devices.

☞ The remote system upgrade feature is not supported in a multi-device chain.

Functional Description

The dedicated remote system upgrade circuitry in Stratix IV devices manages remote configuration and provides error detection, recovery, and status information. User logic or a Nios II processor implemented in the Stratix IV device logic array provides access to the remote configuration data source and an interface to the system's configuration memory.

Stratix IV devices have remote system upgrade processes that involve the following steps:

1. A Nios II processor (or user logic) implemented in the Stratix IV device logic array receives new configuration data from a remote location. The connection to the remote source uses a communication protocol such as the transmission control protocol/Internet protocol (TCP/IP), peripheral component interconnect (PCI), user datagram protocol (UDP), universal asynchronous receiver/transmitter (UART), or a proprietary interface.
2. The Nios II processor (or user logic) stores this new configuration data in non-volatile configuration memory.
3. The Nios II processor (or user logic) initiates a reconfiguration cycle with the new or updated configuration data.
4. The dedicated remote system upgrade circuitry detects and recovers from any error(s) that might occur during or after the reconfiguration cycle and provides error status information to the user design.

Figure 10–21 shows the steps required for performing remote configuration updates. (The numbers in Figure 10–21 coincide with the steps just mentioned.)

Figure 10–21. Functional Diagram of Stratix IV Remote System Upgrade

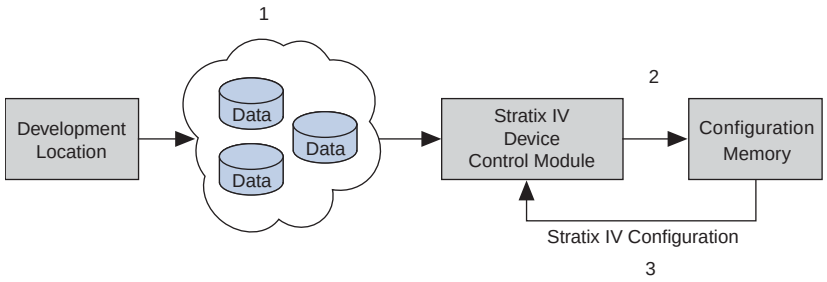
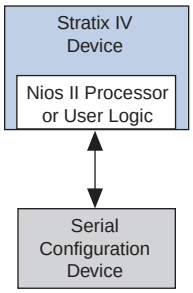


Figure 10–22 shows a block diagram for implementing a remote system upgrade with the Stratix IV fast AS configuration scheme.

Figure 10–22. Remote System Upgrade Block Diagram for Stratix IV Fast AS Configuration Scheme



You must set the mode select pins (MSEL[2..0]) to fast AS mode to use remote system upgrade in your system. Table 10–13 lists the MSEL pin settings for Stratix IV devices in standard configuration mode and remote system upgrade mode. The following sections describe remote update of the remote system upgrade mode.

For more information about standard configuration schemes supported in Stratix IV devices, refer to "Configuration Schemes" on page 10–2.

Table 10–13. Remote System Upgrade Modes in Stratix IV Devices

Configuration Scheme	MSEL[2..0]	Remote System Upgrade Mode
Fast AS (40 MHz)	011	Standard
	011	Remote update *(1)*

Note to Table 10–13:

(1) All EPCS densities are able to support DCLK up to 40 MHz, but batches of EPCS1 and EPCS4 manufactured on 0.18-µm process geometry can only support DCLK up to 20 MHz. For more information, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

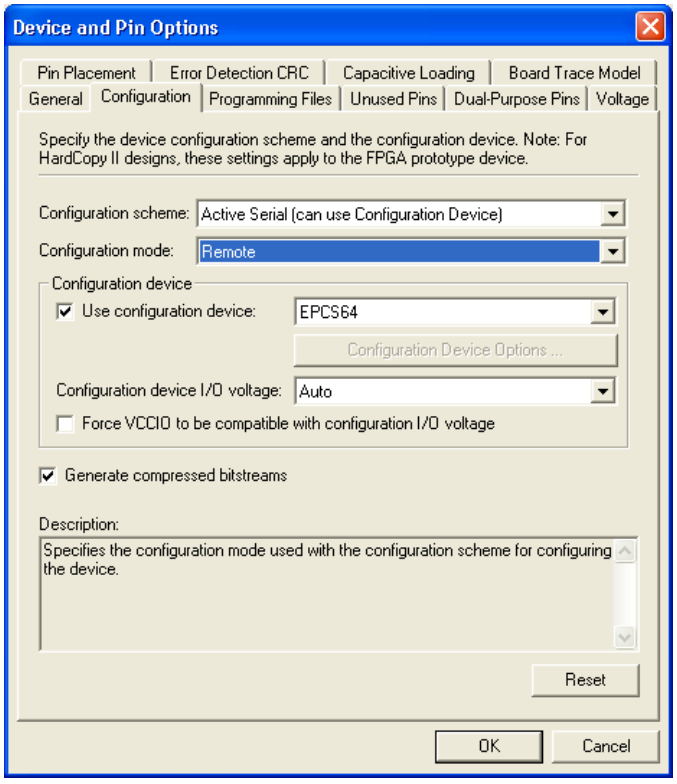
When using fast AS mode, you must select remote update mode in the Quartus II software and insert the ALTREMOTE_UPDATE megafunction to access the circuitry. For more information, refer to "ALTREMOTE_UPDATE Megafunction" on page 10–62.

Enabling Remote Update

You can enable remote update for Stratix IV devices in the Quartus II software before design compilation (in the Compiler Settings menu). In remote update mode, the **auto-restart configuration after error** option is always enabled. To enable remote update in the project's compiler settings, in the Quartus II software, follow these steps:

1. On the Assignment menu, click **Device**. The **Settings** dialog box appears.
2. Click **Device and Pin Options**. The **Device and Pin Options** dialog box appears.
3. Click the **Configuration** tab.
4. From the **Configuration scheme** list, select **Active Serial** (you can also use **Configuration Device**) (Figure 10–23).
5. From the **Configuration Mode** list, select **Remote** (Figure 10–23).
6. Click **OK**.
7. In the **Settings** dialog box, click **OK**.

Figure 10–23. Enabling Remote Update for Stratix IV Devices in the Compiler Settings Menu



Configuration Image Types

When performing a remote system upgrade, Stratix IV device configuration bitstreams are classified as factory configuration images or application configuration images. An image, also referred to as a configuration, is a design loaded into the Stratix IV device that performs certain user-defined functions.

Each Stratix IV device in your system requires one factory image or the addition of one or more application images. The factory image is a user-defined fall-back, or safe configuration, and is responsible for administering remote updates in conjunction with the dedicated circuitry. Application images implement user-defined functionality in the target Stratix IV device. You may include the default application image functionality in the factory image.

A remote system upgrade involves storing a new application configuration image or updating an existing one using the remote communication interface. After an application configuration image is stored or updated remotely, the user design in the Stratix IV device initiates a reconfiguration cycle with the new image. Any errors during or after this cycle are detected by the dedicated remote system upgrade circuitry and cause the device to automatically revert to the factory image. The factory image then performs error processing and recovery. The factory configuration is written to the serial configuration device only once by the system manufacturer and must not be remotely updated. On the other hand, application configurations may be remotely updated in the system. Both images can initiate system reconfiguration.

Remote System Upgrade Mode

Remote system upgrade has one mode of operation—remote update mode. Remote update mode allows you to determine the functionality of your system after power-up and offers several features.

Remote Update Mode

In remote update mode, Stratix IV devices load the factory configuration image after power up. The user-defined factory configuration determines which application configuration is to be loaded and triggers a reconfiguration cycle. The factory configuration may also contain application logic.

When used with serial configuration devices, remote update mode allows an application configuration to start at any flash sector boundary. For example, this translates to a maximum of 128 sectors in the EPCS64 device and 32 sectors in the EPCS16 device, where the minimum size of each page is 512 KBits. Altera recommends not using the same page in the serial configuration devices for two images. Additionally, remote update mode features a user watchdog timer that determines the validity of an application configuration.

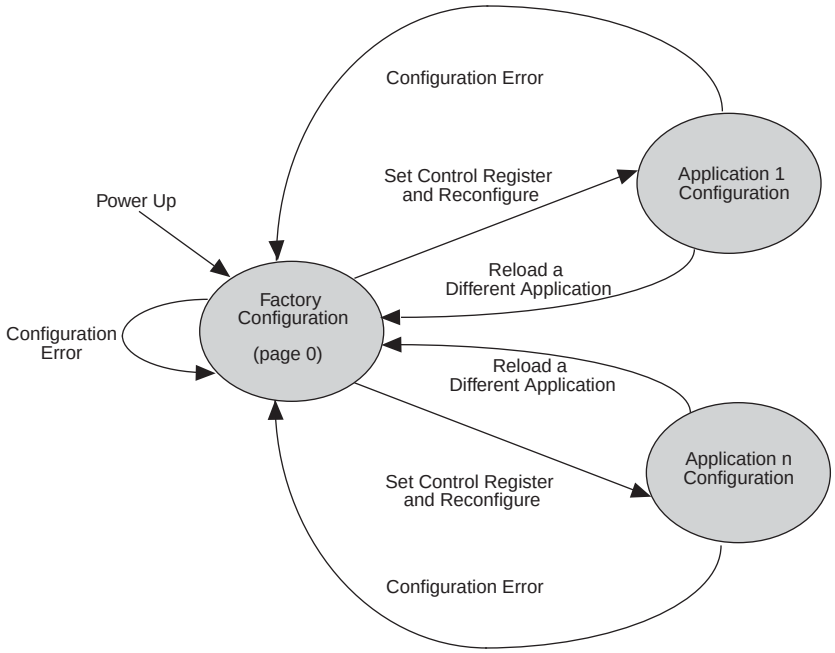
When a Stratix IV device is first powered up in remote update mode, it loads the factory configuration located at page zero (page registers `PGM[23:0] = 24'b0`). Always store the factory configuration image for your system at page address zero. This corresponds to the start address location 0×000000 in the serial configuration device.

The factory image is user-designed and contains soft logic to:

- Process any errors based on status information from the dedicated remote system upgrade circuitry
- Communicate with the remote host and receive new application configurations and store this new configuration data in the local non-volatile memory device
- Determine which application configuration is to be loaded into the Stratix IV device
- Enable or disable the user watchdog timer and load its time-out value (optional)
- Instruct the dedicated remote system upgrade circuitry to initiate a reconfiguration cycle

Figure 10–24 shows the transitions between the factory and application configurations in remote update mode.

Figure 10–24. Transitions Between Configurations in Remote Update Mode



After power up or a configuration error, the factory configuration logic is loaded automatically. The factory configuration also must specify whether to enable the user watchdog timer for the application configuration and if enabled, to include the timer setting information.

The user watchdog timer ensures that the application configuration is valid and functional. The timer must be continually reset within a specific amount of time during user mode operation of an application configuration. Only valid application configurations contain the logic to reset the timer in user mode. This timer reset logic must be part of a user-designed hardware and/or software health monitoring signal that indicates error-free system operation. If the timer is not reset in a specific amount of time; for example, the user application configuration detects a functional problem or if the system hangs, the dedicated circuitry updates the remote system upgrade status register, triggering the loading of the factory configuration.

The user watchdog timer is automatically disabled for factory configurations. For more information about the user watchdog timer, refer to "User Watchdog Timer" on page 10–61.

If there is an error while loading the application configuration, the cause of the reconfiguration is written by the dedicated circuitry to the remote system upgrade status register. Actions that cause the remote system upgrade status register to be written are:

- nSTATUS driven low externally
- Internal CRC error
- User watchdog timer time-out
- A configuration reset (logic array nCONFIG signal or external nCONFIG pin assertion to low)

Stratix IV devices automatically load the factory configuration located at page address zero. This user-designed factory configuration can read the remote system upgrade status register to determine the reason for the reconfiguration. The factory configuration then takes appropriate error recovery steps and writes to the remote system upgrade control register to determine the next application configuration to be loaded.

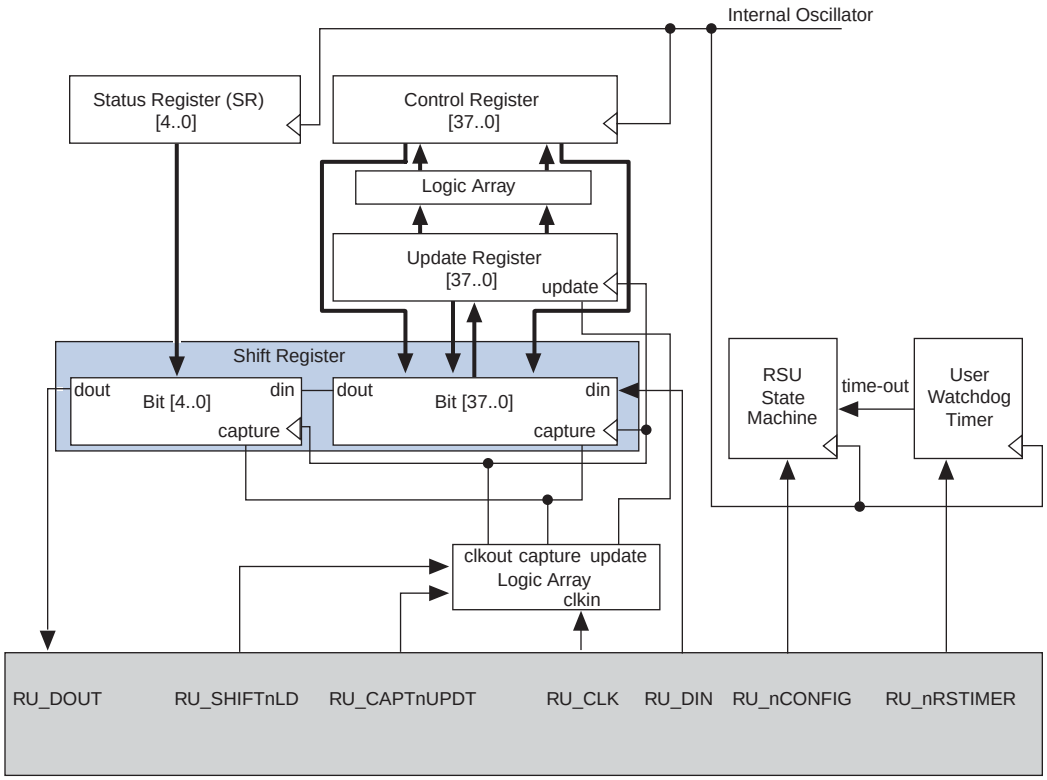
When Stratix IV devices successfully load the application configuration, they enter into user mode. In user mode, the soft logic (Nios II processor or state machine and the remote communication interface) assists the Stratix IV device in determining when a remote system update is arriving. When a remote system update arrives, the soft logic receives the incoming data, writes it to the configuration memory device, and triggers the device to load the factory configuration. The factory configuration reads the remote system upgrade status register and control register, determines the valid application configuration to load, writes the remote system upgrade control register accordingly, and initiates system reconfiguration.

Dedicated Remote System Upgrade Circuitry

This section describes the implementation of the Stratix IV remote system upgrade dedicated circuitry. The remote system upgrade circuitry is implemented in hard logic. This dedicated circuitry interfaces to the user-defined factory and application configurations implemented in the Stratix IV device logic array to provide the complete remote configuration solution. The remote system upgrade circuitry contains the remote system upgrade registers, a watchdog timer, and a state machine that controls those components.

Figure 10–25 shows the data path for the remote system upgrade block.

Figure 10–25. Remote System Upgrade Circuit Data Path ***(Note 1)***



Note to Figure 10–25:

(1) The RU_DOUT, RU_SHIFTnLD, RU_CAPTnUPDT, RU_CLK, RU_DIN, RU_nCONFIG, and RU_nRSTIMER signals are internally controlled by the ALTREMOTE_UPDATE megafunction.

Remote System Upgrade Registers

The remote system upgrade block contains a series of registers that store the page addresses, watchdog timer settings, and status information. Table 10–14 lists these registers.

Table 10–14. Remote System Upgrade Registers

Register	Description
Shift register	This register is accessible by the logic array and allows the update, status, and control registers to be written and sampled by user logic.
Control register	This register contains the current page address, user watchdog timer settings, and one bit specifying whether the current configuration is a factory configuration or an application configuration. During a read operation in an application configuration, this register is read into the shift register. When a reconfiguration cycle is initiated, the contents of the update register are written into the control register.
Update register	This register contains data similar to that in the control register. However, it can only be updated by the factory configuration by shifting data into the shift register and issuing an update operation. When a reconfiguration cycle is triggered by the factory configuration, the control register is updated with the contents of the update register. During a capture in a factory configuration, this register is read into the shift register.
Status register	This register is written to by the remote system upgrade circuitry on every reconfiguration to record the cause of the reconfiguration. This information is used by the factory configuration to determine the appropriate action following a reconfiguration. During a capture cycle, this register is read into the shift register.

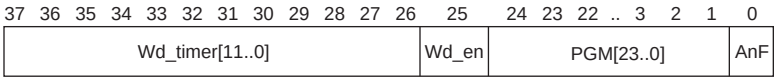
The remote system upgrade control and status registers are clocked by the 10-MHz internal oscillator (the same oscillator that controls the user watchdog timer). However, the remote system upgrade shift and update registers are clocked by the user clock input (RU_CLK).

Remote System Upgrade Control Register

The remote system upgrade control register stores the application configuration page address and user watchdog timer settings. The control register functionality depends on the remote system upgrade mode selection. In remote update mode, the control register page address bits are set to all zeros (24'b0 = 0×000000) at power up to load the factory configuration. A factory configuration in remote update mode has write access to this register.

Figure 10–26 and Table 10–15 specify the control register bit positions. In the figure, the numbers show the bit position of a setting within a register. For example, bit number 25 is the enable bit for the watchdog timer.

Figure 10–26. Remote System Upgrade Control Register



The application-not-factory (AnF) bit indicates whether the current configuration loaded in the Stratix IV device is the factory configuration or an application configuration. This bit is set low by the remote system upgrade circuitry when an error condition causes a fall-back to the factory configuration. When the AnF bit is high, the control register access is limited to read operations. When the AnF bit is low, the register allows write operations and disables the watchdog timer.

In remote update mode, the factory configuration design sets this bit high (1'b1) when updating the contents of the update register with the application page address and watchdog timer settings.

Table 10–15 lists the remote system upgrade control register contents.

Table 10–15. Remote System Upgrade Control Register Contents

Control Register Bit	Remote System Upgrade Mode	Value *(2)*	Definition
AnF *(1)*	Remote update	1'b0	Application not factory
PGM[23..0]	Remote update	24'b0×000000	AS configuration start address (StAdd[23..0])
Wd_en	Remote update	1'b0	User watchdog timer enable bit
Wd_timer[11..0]	Remote update	12'b000000000000	User watchdog time-out value (most significant 12 bits of 29-bit count value: {Wd_timer[11..0], 17'b0})

Notes to Table 10–15:

(1) In remote update mode, the remote configuration block does not update the AnF bit automatically (you can update it manually).

(2) This is the default value of the control register bit.

Remote System Upgrade Status Register

The remote system upgrade status register specifies the reconfiguration trigger condition. The various trigger and error conditions include:

- Cyclic redundancy check (CRC) error during application configuration
- nSTATUS assertion by an external device due to an error
- Stratix IV device logic array triggered a reconfiguration cycle, possibly after downloading a new application configuration image
- External configuration reset (nCONFIG) assertion
- User watchdog timer time-out

Figure 10–27 and Table 10–16 specify the contents of the status register. The numbers in the figure show the bit positions within a 5-bit register.

Figure 10–27. Remote System Upgrade Status Register

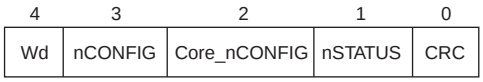


Table 10–16. Remote System Upgrade Status Register Contents

Status Register Bit	Definition	POR Reset Value
CRC (from the configuration)	CRC error caused reconfiguration	1 bit '0'
nSTATUS	nSTATUS caused reconfiguration	1 bit '0'
CORE_nCONFIG *(1)*	Device logic array caused reconfiguration	1 bit '0'
nCONFIG	nCONFIG caused reconfiguration	1 bit '0'
Wd	Watchdog timer caused reconfiguration	1 bit '0'

Note to Table 10–16:

(1) Logic array reconfiguration forces the system to load the application configuration data into the Stratix IV device. This occurs after the factory configuration specifies the appropriate application configuration page address by updating the update register.

Remote System Upgrade State Machine

The remote system upgrade control and update registers have identical bit definitions, but serve different roles (refer to Table 10–14 on page 10–57). While both registers can only be updated when the device is loaded with a factory configuration image, the update register writes are controlled by the user logic; the control register writes are controlled by the remote system upgrade state machine.

In factory configurations, the user logic sends the AnF bit (set high), the page address, and the watchdog timer settings for the next application configuration bit to the update register. When the logic array configuration reset (RU_nCONFIG) goes low, the remote system upgrade state machine updates the control register with the contents of the update register and initiates system reconfiguration from the new application page.

To ensure successful reconfiguration between the pages, assert the RU_nCONFIG signal for a minimum of 250 ns. This is equivalent to strobing the reconfiguration input of the ALTREMOTE_UPDATE megafunction high for a minimum of 250 ns.

In the event of an error or reconfiguration trigger condition, the remote system upgrade state machine directs the system to load a factory or application configuration (page zero or page one, based on the mode and error condition) by setting the control register accordingly. Table 10–17 lists the contents of the control register after such an event occurs for all possible error or trigger conditions.

The remote system upgrade status register is updated by the dedicated error monitoring circuitry after an error condition but before the factory configuration is loaded.

Table 10–17. Control Register Contents after an Error or Reconfiguration Trigger Condition

Reconfiguration Error/Trigger	Control Register Setting Remote Update
nCONFIG reset	All bits are 0
nSTATUS error	All bits are 0
CORE triggered reconfiguration	Update register
CRC error	All bits are 0
Wd time out	All bits are 0

Capture operations during factory configuration access the contents of the update register. This feature is used by the user logic to verify that the page address and watchdog timer settings were written correctly. Read operations in application configurations access the contents of the control register. This information is used by the user logic in the application configuration.

User Watchdog Timer

The user watchdog timer prevents a faulty application configuration from stalling the device indefinitely. The system uses the timer to detect functional errors after an application configuration is successfully loaded into the Stratix IV device.

The user watchdog timer is a counter that counts down from the initial value loaded into the remote system upgrade control register by the factory configuration. The counter is 29 bits wide and has a maximum count value of 2^{29}. When specifying the user watchdog timer value, specify only the most significant 12 bits. The granularity of the timer setting is 2^{17} cycles. The cycle time is based on the frequency of the 10-MHz internal oscillator. Table 10–18 lists the operating range of the 10-MHz internal oscillator.

Table 10–18. 10-MHz Internal Oscillator Specifications ***(Note 1)***

Minimum	Typical	Maximum	Units
4.3	5.3	10	MHz

Note to Table 10–18:

(1) These values are preliminary.

The user watchdog timer begins counting after the application configuration enters device user mode. This timer must be periodically reloaded or reset by the application configuration before the timer expires by asserting RU_nRSTIMER. If the application configuration does not reload the user watchdog timer before the count expires, a time-out signal is generated by the remote system upgrade dedicated circuitry. The time-out signal tells the remote system upgrade circuitry to set the user watchdog timer status bit (Wd) in the remote system upgrade status register and reconfigures the device by loading the factory configuration.

To allow remote system upgrade dedicated circuitry to reset the watchdog timer, you must assert the RU_nRSTIMER signal active for a minimum of 250 ns. This is equivalent to strobing the reset_timer input of the ALTREMOTE_UPDATE megafunction high for a minimum of 250 ns.

The user watchdog timer is not enabled during the configuration cycle of the device. Errors during configuration are detected by the CRC engine. Also, the timer is disabled for factory configurations. Functional errors should not exist in the factory configuration because it is stored and validated during production and is never updated remotely.

The user watchdog timer is disabled in factory configurations and during the configuration cycle of the application configuration. It is enabled after the application configuration enters user mode.

Quartus II Software Support

The Quartus II software provides the flexibility to include the remote system upgrade interface between the Stratix IV device logic array and the dedicated circuitry, generate configuration files for production, and allows remote programming of the system configuration memory.

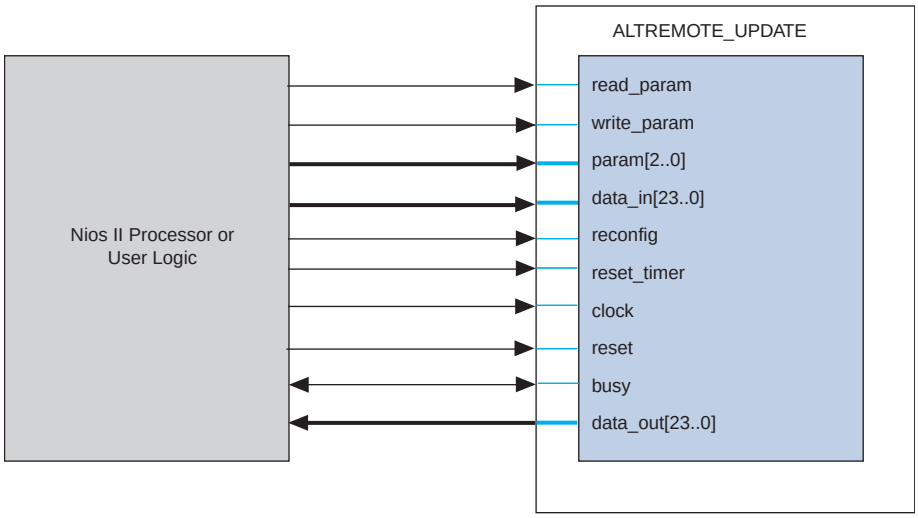
The ALTREMOTE_UPDATE megafunction is the implementation option in the Quartus II software that you use for the interface between the remote system upgrade circuitry and the device logic array interface. Using the megafunction block instead of creating your own logic saves design time and offers more efficient logic synthesis and device implementation.

ALTREMOTE_UPDATE Megafunction

The ALTREMOTE_UPDATE megafunction provides a memory-like interface to the remote system upgrade circuitry and handles the shift register read and write protocol in the Stratix IV device logic. This implementation is suitable for designs that implement the factory configuration functions using a Nios II processor or user logic in the device.

Figure 10–28 shows the interface signals between the ALTREMOTE_UPDATE megafunction and Nios II processor or user logic.

Figure 10–28. Interface Signals between the ALTREMOTE_UPDATE Megafunction and the Nios II Processor



For more information about the ALTREMOTE_UPDATE megafunction and the description of ports shown in Figure 10–28, refer to the *Remote Update Circuitry (ALTREMOTE_UPDATE) Megafunction User Guide*.

Design Security

This section provides an overview of the design security feature and its implementation on Stratix IV devices using the advanced encryption standard (AES). It also covers the new security modes available in Stratix IV devices.

As Stratix IV devices continue play a role in larger and more critical designs in competitive commercial and military environments, it is increasingly important to protect the designs from copying, reverse engineering, and tampering.

Stratix IV devices address these concerns with both volatile and non-volatile security feature support. Stratix IV devices have the ability to decrypt configuration bitstreams using the AES algorithm, an industry-standard encryption algorithm that is FIPS-197 certified. Stratix IV devices have a design security feature that utilizes a 256-bit security key.

Stratix IV devices store configuration data in SRAM configuration cells during device operation. Because SRAM is volatile, the SRAM cells must be loaded with configuration data each time the device powers up. It is possible to intercept configuration data when it is being transmitted from the memory source (flash memory or a configuration device) to the device. The intercepted configuration data could then be used to configure another device.

When using the Stratix IV design security feature, the security key is stored in the Stratix IV device. Depending on the security mode, you can configure the Stratix IV device using a configuration file that is encrypted with the same key, or for board testing, configured with a normal configuration file.

The design security feature is available when configuring Stratix IV devices using FPP configuration mode with an external host (such as a MAX II device or microprocessor), or when using fast AS or PS configuration schemes. The design security feature is also available in remote update with fast AS configuration mode. The design security feature is not available when you are configuring your Stratix IV device using JTAG-based configuration. For more information, refer to "Supported Configuration Schemes" on page 10–67.

When using a serial configuration scheme such as PS or fast AS, configuration time is the same whether or not you enable the design security feature. If the FPP scheme is used with the design security or decompression feature, a ×4 DCLK is required. This results in a slower configuration time when compared with the configuration time of a Stratix IV device that has neither the design security nor the decompression feature enabled.

Stratix IV Security Protection

Stratix IV device designs are protected from copying, reverse engineering, and tampering using configuration bitstream encryption.

Security Against Copying

The security key is securely stored in the Stratix IV device and cannot be read out through any interfaces. In addition, as configuration file read-back is not supported in Stratix IV devices, the design information cannot be copied.

Security Against Reverse Engineering

Reverse engineering from an encrypted configuration file is very difficult and time consuming because the Stratix IV configuration file formats are proprietary and the file contains millions of bits which require specific decryption. Reverse engineering the Stratix IV device is just as difficult because the device is manufactured on the most advanced 40-nm process technology.

Security Against Tampering

The non-volatile keys are one-time programmable. After the Tamper Protection bit is set in the key programming file generated by the Quartus II software, the Stratix IV device can only be configured with configuration files encrypted with the same key.

AES Decryption Block

The main purpose of the AES decryption block is to decrypt the configuration bitstream prior to entering data decompression or configuration.

Prior to receiving encrypted data, you must enter and store the 256-bit security key in the device. You can choose between a non-volatile security key and a volatile security key with battery backup.

The security key is scrambled prior to storing it in the key storage to make it more difficult for anyone to retrieve the stored key using de-capsulation of the device.

Flexible Security Key Storage

Stratix IV devices support two types of security key programming—volatile and non-volatile keys. Table 10–19 lists the differences between volatile keys and non-volatile keys.

Table 10–19. Security Key Options (Part 1 of 2)

Options	Volatile Key	Non-Volatile Key
Key programmability	Reprogrammable and erasable	One-time programmable
External battery	Required	Not required
Key programming method *(1)*	On-board	On and off board

Table 10–19. Security Key Options (Part 2 of 2)

Options	Volatile Key	Non-Volatile Key
Design protection	Secure against copying and reverse engineering	Secure against copying and reverse engineering. Tamper resistant if tamper protection bit is set.

Note to Table 10–19:

(1) Key programming is carried out using the JTAG interface.

You can program the non-volatile key to the Stratix IV device without an external battery. Also, there are no additional requirements to any of the Stratix IV power supply inputs.

V_{CCBAT} is a dedicated power supply for volatile key storage and not shared with other on-chip power supplies, such as V_{CCIO} or V_{CC}. V_{CCBAT} continuously supplies power to the volatile register regardless of the on-chip supply condition.

After power-up, you must wait 300 ms (PORSEL = 0) or 12 ms (PORSEL = 1) before beginning key programming to ensure that V_{CCBAT} is at full rail.

For more information about how to calculate the key retention time of the battery used for volatile key storage, refer to the *Stratix III, Stratix IV, Stratix V, HardCopy III and HardCopy IV PowerPlay Early Power Estimator*.

For more information about battery specifications, refer to the *DC and Switching Characteristics in Stratix IV Devices* chapter.

For more information about the V_{CCBAT} pin connection recommendations, refer to the *Stratix IV GX and Stratix IV E Device Family Pin Connection Guidelines*.

Stratix IV Design Security Solution

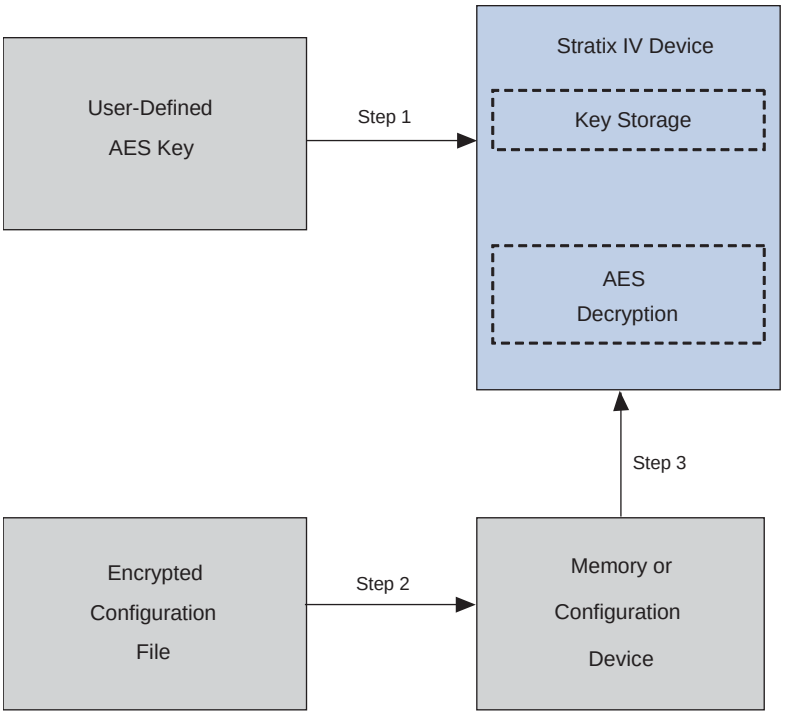
Stratix IV devices are SRAM-based devices. To provide design security, Stratix IV devices require a 256-bit security key for configuration bitstream encryption.

You can carry out secure configuration in the following steps, as shown in Figure 10–29:

1. Program the security key into the Stratix IV device.
2. Program the user-defined 256-bit AES keys to the Stratix IV device through the JTAG interface.
3. Encrypt the configuration file and store it in the external memory.
4. Encrypt the configuration file with the same 256-bit keys used to program the Stratix IV device. Encryption of the configuration file is done using the Quartus II software. The encrypted configuration file is then loaded into the external memory, such as a configuration or flash device.
5. Configure the Stratix IV device.

At system power-up, the external memory device sends the encrypted configuration file to the Stratix IV device.

Figure 10–29. Design Security ***(Note 1)***



Note to Figure 10–29:

(1) Step 1, Step 2, and Step 3 correspond to the procedure described in "Design Security" on page 10–63.

Security Modes Available

The following security modes are available on the Stratix IV device.

Volatile Key

Secure operation with volatile key programmed and required external battery: this mode accepts both encrypted and unencrypted configuration bitstreams. Use the unencrypted configuration bitstream support for board-level testing only.

Non-Volatile Key

Secure operation with one time programmable (OTP) security key programmed: this mode accepts both encrypted and unencrypted configuration bitstreams. Use the unencrypted configuration bitstream support for board level testing only.

Non-Volatile Key with Tamper Protection Bit Set

Secure operation in tamper resistant mode with OTP security key programmed: only encrypted configuration bitstreams are allowed to configure the device. Tamper protection disables JTAG configuration with unencrypted configuration bitstream.

Enabling the tamper protection bit disables test mode in Stratix IV devices. This process is irreversible and prevents Altera from conducting carry-out failure analysis if test mode is disabled. Contact Altera Technical Support to enable the tamper protection bit.

No Key Operation

Only unencrypted configuration bitstreams are allowed to configure the device.

Table 10–20 lists the different security modes and configuration bitstream supported for each mode.

Table 10–20. Security Modes Supported

Mode *(1)*	Function	Configuration File
Volatile key	Secure	Encrypted
	Board-level testing	Unencrypted
Non-volatile key	Secure	Encrypted
	Board-level testing	Unencrypted
Non-volatile key with tamper protection bit set	Secure (tamper resistant) *(2)*	Encrypted

Notes to Table 10–20:

(1) In No key operation, only the unencrypted configuration file is supported.

(2) The tamper protection bit setting does not prevent the device from being reconfigured.

Supported Configuration Schemes

The Stratix IV device supports only selected configuration schemes, depending on the security mode you select when you encrypt the Stratix IV device.

Figure 10–30 shows the restrictions of each security mode when encrypting Stratix IV devices.

Figure 10–30. Security Modes in Stratix IV Devices—Sequence and Restrictions

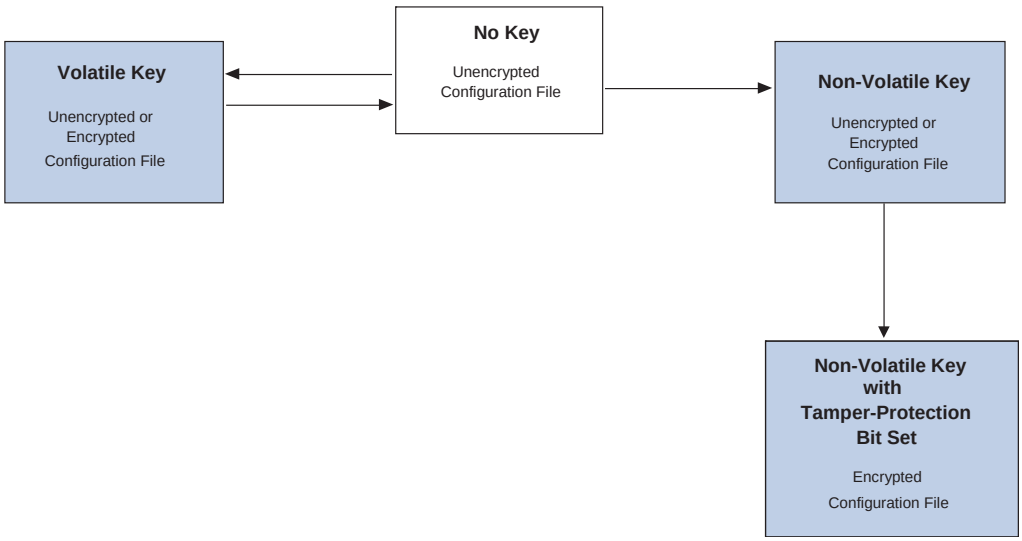


Table 10–21 lists the configuration modes allowed in each of the security modes.

Table 10–21. Allowed Configuration Modes for Various Security Modes ***(Note 1)***

Security Mode	Configuration File	Allowed Configuration Modes
No key	Unencrypted	All configuration modes that do not engage the design security feature.
Secure with volatile key	Encrypted	■ Passive serial with AES (and/or with decompression) ■ Fast passive parallel with AES (and/or with decompression) ■ Remote update fast AS with AES (and/or with decompression) ■ Fast AS (and/or with decompression)
Board-level testing with volatile key	Unencrypted	All configuration modes that do not engage the design security feature.
Secure with non-volatile key	Encrypted	■ Passive serial with AES (and/or with decompression) ■ Fast passive parallel with AES (and/or with decompression) ■ Remote update fast AS with AES (and/or with decompression) ■ Fast AS (and/or with decompression)
Board-level testing with non-volatile key	Unencrypted	All configuration modes that do not engage the design security feature.
Secure in tamper resistant mode using non-volatile key with tamper protection set	Encrypted	■ Passive serial with AES (and/or with decompression) ■ Fast passive parallel with AES (and/or with decompression) ■ Remote update fast AS with AES (and/or with decompression) ■ Fast AS (and/or with decompression)

Note to Table 10–21:

(1) There is no impact to the configuration time required when compared with unencrypted configuration modes except FPP with AES (and/or decompression), which requires a DCLK that is ×4 the data rate.

You can use the design security feature with other configuration features, such as compression and remote system upgrade features. When you use compression with the design security feature, the configuration file is first compressed and then encrypted using the Quartus II software. During configuration, the Stratix IV device first decrypts and then decompresses the configuration file.

Document Revision History

Table 10–22 lists the revision history for this chapter.

Table 10–22. Document Revision History (Part 1 of 2)

Date	Version	Changes
April 2011	3.3	■ Updated the "FPP Configuration Using a MAX II Device as an External Host", "Fast Active Serial Configuration (Serial Configuration Devices)", and "PS Configuration Using a MAX II Device as an External Host". ■ Updated Table 10–10.
February 2011	3.2	■ Updated the "Fast Active Serial Configuration (Serial Configuration Devices)", "FPP Configuration Using a MAX II Device as an External Host" "Configuration Data Decompression", and "User Watchdog Timer" sections. ■ Updated Table 10–2, Table 10–4, Table 10–5, Table 10–7, and Table 10–9. ■ Applied new template. ■ Minor text edits.
March 2010	3.1	■ Added the "Guidelines for Connecting Serial Configuration Devices on an AS Interface" section. ■ Updated the "Power-On Reset Circuit" and "Fast Active Serial Configuration (Serial Configuration Devices)" sections. ■ Updated Table 10–2, Table 10–4, Table 10–5, Table 10–10, and Table 10–13. ■ Updated Figure 10–16 and Figure 10–17 with Note 5. ■ Updated Figure 10–4, Figure 10–5, and Figure 10–13. ■ Updated the reference in the "Configuration Schemes" section.
November 2009	3.0	■ Updated Table 10–1 and Table 10–2. ■ Updated the "FPP Configuration Using a MAX II Device as an External Host","Fast Active Serial Configuration (Serial Configuration Devices)", "Device Configuration Pins", "Remote System Upgrades", "Remote System Upgrade Mode", "Estimating Active Serial Configuration Time", "Remote System Upgrade State Machine", and "User Watchdog Timer" sections. ■ Removed Table 10-4, Table 10-7, Table 10-8, and Table 10-25. ■ Minor text edits.
June 2009	2.3	■ Updated the "VCCPD Pins", "FPP Configuration Using a MAX II Device as an External Host", "Estimating Active Serial Configuration Time", "Fast Active Serial Configuration (Serial Configuration Devices)", "Remote System Upgrades", "PS Configuration Using a MAX II Device as an External Host", and "PS Configuration Using a Download Cable" sections. ■ Updated Table 10–3, Table 10–13 and Table 10–2. ■ Added introductory sentences to improve search ability. ■ Removed the Conclusion section. ■ Minor text edits.
April 2009	2.2	■ Updated Table 10–2.
March 2009	2.1	■ Updated Table 10–1, Table 10–2, and Table 10–9. ■ Removed "Referenced Documents" section.

Table 10–22. Document Revision History (Part 2 of 2)

Date	Version	Changes
November 2008	2.0	■ Updated "Fast Active Serial Configuration (Serial Configuration Devices)" and "JTAG Configuration" sections. ■ Updated Figure 10–4, Figure 10–5, Figure 10–6, and Figure 10–13. ■ Updated Table 10–2 and Table 10–13.
May 2008	1.0	Initial release.

11. Configuring Stratix III Devices

SIII51011-2.0

This chapter contains complete information about Stratix® III supported configuration schemes, how to execute the required configuration schemes, and all necessary option pin settings.

Stratix III devices use SRAM cells to store configuration data. Because SRAM memory is volatile, you must download configuration data to the Stratix III device each time the device powers up. You can configure Stratix III devices using one of four configuration schemes:

- Fast passive parallel (FPP)
- Fast active serial (AS)
- Passive serial (PS)
- Joint Test Action Group (JTAG)

All configuration schemes use an external controller (for example, a MAX® II device or microprocessor), a configuration device, or a download cable. Refer to "Configuration Features" on page 11–3 for more information.

Configuration Devices

The Altera® serial configuration devices (EPCS128, EPCS64, and EPCS16) support a single-device and multi-device configuration solution for Stratix III devices and are used in the fast AS configuration scheme. Serial configuration devices offer a low-cost, low-pin count configuration solution.

For information about serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook*.

All minimum timing information in this handbook covers the entire Stratix III family. Some devices may work at less than the minimum timing stated in this handbook due to process variation.

Configuration Schemes

Select the configuration scheme by driving the Stratix III device `MSEL` pins either high or low, as detailed in Table 11–1. The `MSEL` pins are powered by the V_{CCPGM} power supply of the bank they reside in. The `MSEL[2..0]` pins have 5-kΩ internal pull-down resistors that are always active. During power-on reset (POR) and reconfiguration, the `MSEL` pins must be at LVTTL V_{IL} and V_{IH} levels to be considered a logic low and logic high.

To avoid any problems with detecting an incorrect configuration scheme, hard-wire the `MSEL[]` pins to V_{CCPGM} and GND, without any pull-up or pull-down resistors. Do not drive the `MSEL[]` pins with a microprocessor or another device.

Table 11–1. Stratix III Configuration Schemes

Configuration Scheme	MSEL2	MSEL1	MSEL0
Fast passive parallel (FPP)	0	0	0
Passive serial (PS)	0	1	0
Fast AS (40 MHz) *(1)*	0	1	1
Remote system upgrade fast AS (40 MHz) *(1)*	0	1	1
FPP with design security feature, decompression, or both enabled *(2)*	0	0	1
JTAG-based configuration *(4)*	*(3)*	*(3)*	*(3)*

Notes to Table 11–1:

(1) To support fast AS configuration for Stratix III, you must use EPCS16, EPCS64, or EPCS128 devices. For more information, refer to *Serial Configuration Devices Data Sheet* chapter.

(2) These modes are only supported when using a MAX® II device or a microprocessor with flash memory for configuration. In these modes, the host system must output a DCLK that is ×4 the data rate.

(3) Do not leave the MSEL pins floating. Connect them to VCCPGM or ground. These pins support the non-JTAG configuration scheme used in production. If you only use JTAG configuration, connect the MSEL pins to ground.

(4) JTAG-based configuration takes precedence over other configuration schemes, which means MSEL pin settings are ignored.

Table 11–2 lists the uncompressed raw binary file (**.rbf**) configuration file sizes for Stratix III devices.

Table 11–2. Stratix III Uncompressed Raw Binary File (**.rbf**) Sizes

Device	Data Size (Bits)
EP3SL50	22, 178, 792
EP3SL70	22, 178, 792
EP3SL110	47, 413, 312
EP3SL150	47, 413, 312
EP3SL200	93, 324, 656
EP3SL340	117, 387, 664
EP3SE50	25, 891, 968
EP3SE80	48, 225, 392
EP3SE110	48, 225, 392
EP3SE260	93, 324, 656

Use the data in Table 11–2 to estimate the file size before design compilation. Different configuration file formats, such as a hexadecimal (**.hex**) or tabular text file (**.ttf**) format, have different file sizes. Refer to the Quartus® II software for the different types of configuration file and the file sizes. However, for any specific version of the Quartus II software, any design targeted for the same device will have the same uncompressed configuration file size. If you are using compression, the file size can vary after each compilation because the compression ratio is dependent on the design.

For more information about setting device configuration options or creating configuration files, refer to the *Device Configuration Options and Configuration File Formats chapters* in volume 2 of the *Configuration Handbook*.

Configuration Features

Stratix III devices offer design security, decompression, and remote system upgrade features. Design security using configuration bitstream encryption is available in Stratix III devices, which protects your designs. Stratix III devices can receive a compressed configuration bitstream and decompress this data in real-time, reducing storage requirements and configuration time. You can make real-time system upgrades from remote locations of your Stratix III designs with the remote system upgrade feature.

Table 11–3 summarizes which configuration features you can use in each configuration scheme.

Table 11–3. Stratix III Configuration Features

Configuration Scheme	Configuration Method	Decompression	Design Security	Remote System Upgrade
FPP	MAX II device or a microprocessor with flash memory	✓ *(1)*	✓ *(1)*	—
Fast AS	Serial configuration device	✓	✓	✓ *(2)*
PS	MAX II device or a microprocessor with flash memory	✓	✓	—
	Download cable	✓	✓	—
JTAG	MAX II device or a microprocessor with flash memory	—	—	—
	Download cable	—	—	—

Notes to Table 11–3:

(1) In these modes, the host system must send a DCLK that is ×4 the data rate.

(2) Remote system upgrade is only available in the fast AS configuration scheme. Only remote update mode is supported when using the fast AS configuration scheme. Local update mode is not supported.

If your system already contains a common flash interface (CFI) flash memory, you can use it for the Stratix III device configuration storage as well. The MAX II parallel flash loader (PFL) feature in MAX II devices provides an efficient method to program CFI flash memory devices through the JTAG interface, and the logic to control configuration from the flash memory device to the Stratix III device. Both PS and FPP configuration modes are supported using the PFL feature.

For more information about PFL, refer to *AN 386: Using the MAX II Parallel Flash Loader with the Quartus II Software*.

For more information about programming Altera serial configuration devices, refer to "Programming Serial Configuration Devices" on page 11–25.

Configuration Data Decompression

Stratix III devices support configuration data decompression, which saves configuration memory space and time. This feature allows you to store compressed configuration data in configuration devices or other memory and transmit this compressed bitstream to Stratix III devices. During configuration, the Stratix III device decompresses the bitstream in real time and programs its SRAM cells.

Preliminary data indicates that compression typically reduces the configuration bitstream size by 35 to 55%, based on the designs used.

Stratix III devices support decompression in the FPP (when using a MAX II device/microprocessor + flash), fast AS, and PS configuration schemes. The Stratix III decompression feature is not available in the JTAG configuration scheme.

When using FPP mode, the intelligent host must provide a DCLK that is ×4 the data rate. Therefore, the configuration data must be valid for four DCLK cycles.

In PS mode, use the Stratix III decompression feature, because sending compressed configuration data reduces configuration time.

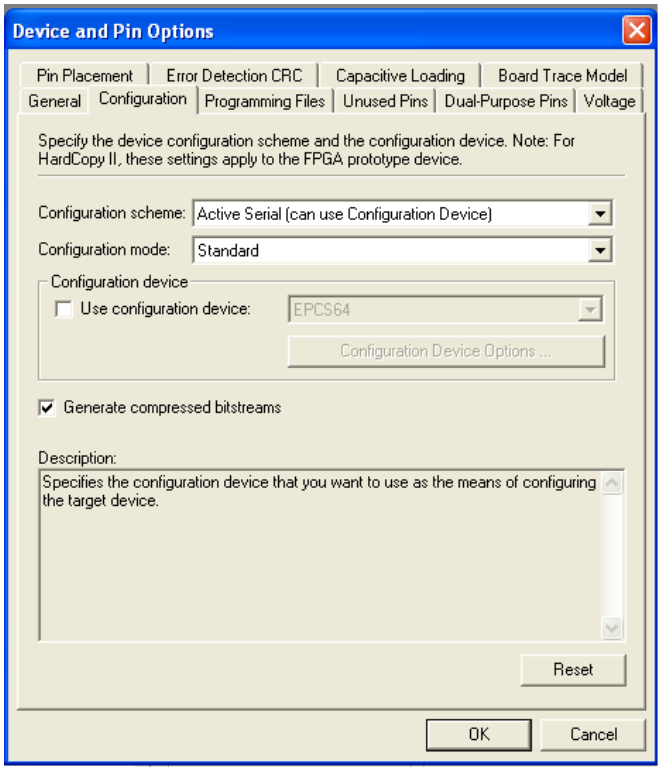
When you enable compression, the Quartus II software generates configuration files with compressed configuration data. This compressed file reduces the storage requirements in the configuration device or flash memory, and decreases the time needed to transmit the bitstream to the Stratix III device. The time required by a Stratix III device to decompress a configuration file is less than the time needed to transmit the configuration data to the device.

There are two ways to enable compression for Stratix III bitstreams: before design compilation (in the Compiler Settings menu) and after design compilation (in the Convert Programming Files window).

To enable compression in the project's Compiler Settings menu, perform the following steps:

1. On the Assignments menu, click **Device**. The **Settings** dialog box appears.
2. In the **Family** list, select **Stratix III** and then click the **Device and Pin Options** button.
3. On the **Configuration** tab, turn on the **Generate compressed bitstreams** option(Figure 11–1).

Figure 11–1. Enabling Compression for Stratix III Bitstreams in Compiler Settings



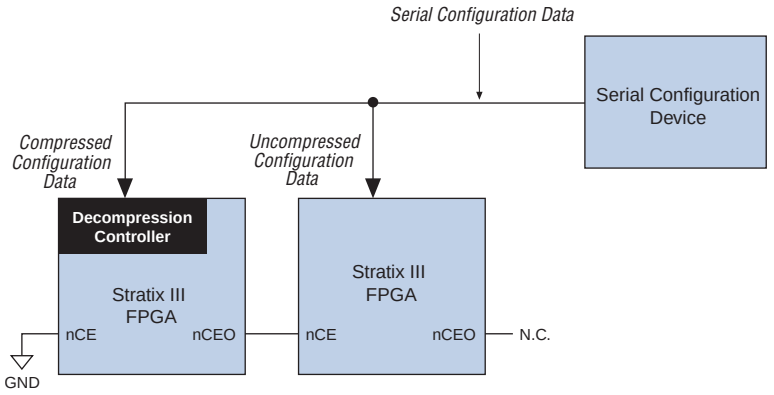
You can also enable compression when creating programming files from the Convert Programming Files window.

1. On the File menu, click **Convert Programming Files**.
2. In the (**.pof**, **.sram**, **.hex**, **.rbf**, or **.ttf**) list, select the programming file type.
3. For POF output files, select a configuration device from (**.pof**, **.sram**, **.hex**, **.rbf**, or **.ttf**).
4. Under **Input files to convert**, select **SOF Data**.
5. Select **Add File** and add a Stratix III device SOF or SOFs.
6. Select the name of the file you added to the **SOF Data** area and click **Properties**.
7. Check the **Compression** check box.

When multiple Stratix III devices are cascaded, you can selectively enable the compression feature for each device in the chain if you are using a serial configuration scheme. Figure 11–2 shows a chain of two Stratix III devices. The first Stratix III device has compression enabled, and receives a compressed bitstream from the configuration device. The second Stratix III device has the compression feature disabled, and receives uncompressed data.

In a multi-device FPP configuration chain (with a MAX II device/microprocessor + flash), all Stratix III devices in the chain must either enable or disable the decompression feature. You cannot selectively enable the compression feature for each device in the chain because of the DATA and DCLK relationship.

Figure 11–2. Compressed and Uncompressed Configuration Data in the Same Configuration File



To generate programming files for this setup in the Quartus II software, on the File menu, click **Convert Programming Files**.

Design Security Using Configuration Bitstream Encryption

Stratix III devices support decryption of configuration bitstreams using the advanced encryption standard (AES) algorithm—the most advanced encryption algorithm available today. Both non-volatile and volatile key programming are supported using Stratix III devices. When using the design security feature, a 256-bit security key is stored in the Stratix III device. To successfully configure a Stratix III device that has the design security feature enabled, the device must be configured with a configuration file that was encrypted using the same 256-bit security key. Non-volatile key programming does not require any external devices, such as a battery backup, for storage. However, for certain applications, you can store the security keys in volatile memory in the Stratix III device. An external battery is needed for this volatile key storage.

When using a serial configuration scheme such as PS or fast AS, configuration time is the same whether or not the design security feature is enabled. If the FPP scheme is used with the design security or decompression feature, a ×4 DCLK is required. This results in a slower configuration time when compared to the configuration time of a Stratix III device that has neither the design security nor the decompression feature enabled.

For more information about this feature, refer to the *Design Security in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*.

Remote System Upgrade

Stratix III devices contain the remote update feature. For more information about this feature, refer to the *Remote System Upgrades with Stratix III Devices* in volume 1 of the *Stratix III Device Handbook*.

Power-On Reset Circuit

The POR circuit keeps the entire system in reset until the power supply voltage levels have stabilized on power-up. On power-up, the device does not release nSTATUS until V_{CCPT}, V_{CCL}, V_{CC}, V_{CCPD}, and V_{CCPGM} are above the device's POR trip point. On power down, brown-out occurs if V_{CC} or V_{CCL} ramps down below the POR trip point and V_{CC}, V_{CCPD}, or V_{CCPGM} drops below the threshold voltage.

In Stratix III devices, a pin-selectable option (PORSEL) is provided that allows you to select a typical POR time setting of 12 ms or 100 ms. In both cases, you can extend the POR time by using an external component to assert the nSTATUS pin low.

V_{CCPGM} Pins

Stratix III devices offer a new power supply, V_{CCPGM}, for all the dedicated configuration pins and dual function pins. The configuration voltages supported are 1.8 V, 2.5 V, 3.0 V, and 3.3 V. Stratix III devices do not support the 1.5 V configuration.

Use this pin to power all dedicated configuration inputs, dedicated configuration outputs, dedicated configuration bi-directional pins, and some of the dual functional pins that you use for configuration. With V_{CCPGM}, configuration input buffers do not have to share power lines with the regular I/O buffer in Stratix III devices.

The operating voltage for the configuration input pin is independent of the I/O bank's power supply V_{CCIO} during the configuration. Therefore, no configuration voltage constraints on V_{CCIO} are needed in Stratix III devices.

V_{CCPD} Pins

Stratix III devices have a dedicated programming power supply, V_{CCPD}, which must be connected to 3.3 V/3.0 V/2.5 V to power the I/O pre-drivers, the JTAG input and output pins (TCK, TMS, TDI, TDO, and TRST), and the design security circuitry.

V_{CCPGM} and V_{CCPD} must ramp up from 0 V to the desired voltage level within 100 ms. If these supplies are not ramped up within this specified time, your Stratix III device will not configure successfully. If your system does not allow ramp-up time of 100 ms or less, you must hold nCONFIG low until all power supplies are stable.

For more information about the configuration pins power supply, refer to "Device Configuration Pins" on page 11–43.

Fast Passive Parallel Configuration

Fast passive parallel (FPP) configuration in Stratix III devices is designed to meet the continuously increasing demand for faster configuration times. Stratix III devices are designed with the capability of receiving byte-wide configuration data per clock cycle. Table 11–4 lists the MSEL pin settings when using the FPP configuration scheme.

Table 11–4. Stratix III MSEL Pin Settings for FPP Configuration Schemes

Configuration Scheme	MSEL2	MSEL1	MSEL0
Fast Passive Parallel (FPP)	0	0	0
FPP with the design security feature, decompression feature, or both enabled *(1)*	0	0	1

Note to Table 11–4:

(1) These modes are only supported when using a MAX II device or a microprocessor with flash memory for configuration. In these modes, the host system must output a DCLK that is ×4 the data rate.

You can perform FPP configuration of Stratix III devices using an intelligent host, such as a MAX II device, or a microprocessor.

FPP Configuration Using a MAX II Device as an External Host

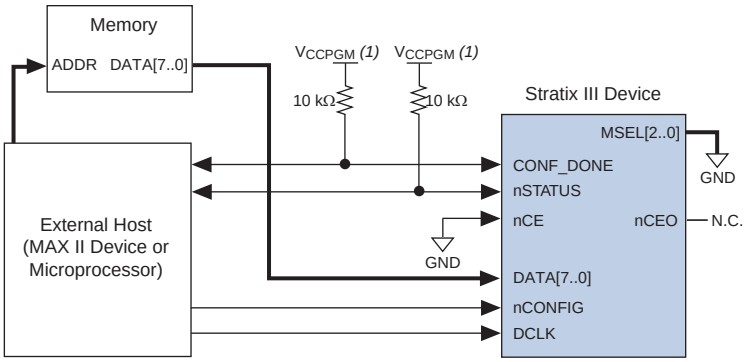
FPP configuration using compression and an external host provides the fastest method to configure Stratix III devices. In this configuration scheme, you can use a MAX II device as an intelligent host that controls the transfer of configuration data from a storage device, such as flash memory, to the target Stratix III device. You can store configuration data in **.rbf**, **.hex**, or **.ttf** format. When using the MAX II device as an intelligent host, a design that controls the configuration process, such as fetching the data from flash memory and sending it to the device, must be stored in the MAX II device.

If you are using the Stratix III decompression feature, design security feature or both, the external host must be able to send a DCLK frequency that is four times the data rate.

The ×4 DCLK signal does not require an additional pin and is sent on the DCLK pin. The maximum DCLK frequency is 100 MHz, which results in a maximum data rate of 200 Mbps. If you are not using the Stratix III decompression or design security features, the data rate is the same as the DCLK frequency.

Figure 11–3 shows the configuration interface connections between the Stratix III device and a MAX II device for single device configuration.

Figure 11–3. Single Device FPP Configuration Using an External Host



Note to Figure 11–3:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Stratix III device. V_{CCPGM} should be high enough to meet the V_{IH} specification of the I/O on the external host. It is recommended to power up all configuration system's I/O with V_{CCPGM}.

Upon power-up, the Stratix III device goes through a POR. The POR delay is dependent on the PORSEL pin setting. When PORSEL is driven low, the POR time is approximately 100 ms. When PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device resets, holds nSTATUS low, and tri-states all user I/O pins. After the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins have weak pull-up resistors, which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS is low, the device is in the reset stage. To initiate configuration, the MAX II device must drive the nCONFIG pin from low to high.

☞ V_{CC}, V_{CCIO}, V_{CCPGM}, and V_{CCPD} of the banks where the configuration and JTAG pins reside must be fully powered to the appropriate voltage levels to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. After nSTATUS is released, the device is ready to receive configuration data and the configuration stage begins. When nSTATUS is pulled high, the MAX II device places the configuration data one byte at a time on the DATA[7..0] pins.

☞ Stratix III devices receive configuration data on the DATA[7..0] pins and the clock is received on the DCLK pin. Data is latched into the device on the rising edge of DCLK. If you are using the Stratix III decompression feature, design security feature, or both, the configuration data is latched on the rising edge of every fourth DCLK cycle. After the configuration data is latched in, it is processed during the following three DCLK cycles.

Data is continuously clocked into the target device until CONF_DONE goes high. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin. The CONF_DONE pin must have an external 10-kΩ pull-up resistor for the device to initialize.

In Stratix III devices, the initialization clock source is either the internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If you use the internal oscillator, the Stratix III device receives enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device. Driving DCLK to the device after configuration is complete does not affect device operation.

You can also synchronize initialization of multiple devices or delay initialization with the CLKUSR option. You can turn on the **Enable user-supplied start-up clock (CLKUSR)** option in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box. Supplying a clock on CLKUSR does not affect the configuration process. After the CONF_DONE pin transitions high, CLKUSR is enabled after the time specified as t_{CD2CU}. When this time period elapses, Stratix III devices require 4,436 clock cycles to initialize properly and enter user mode. Stratix III devices support a CLKUSR f_{MAX} of 100 MHz.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user mode with a low-to-high transition. The **Enable INIT_DONE Output** option is available in the Quartus II software on the **General** tab of the **Device and Pin Options** dialog box. If you use the INIT_DONE pin, it is high because of an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. After the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. The MAX II device must be able to detect this low-to-high transition, which signals the device has entered user mode. When initialization is complete, the device enters user mode. In user mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

To ensure DCLK and DATA[7..0] are not left floating at the end of configuration, the MAX II device must drive them either high or low, whichever is convenient on your board. During configuration, DATA[7..0] pins are powered by V_{CCPGM}. After entering user mode, these pins are available as user I/O pins that are powered by V_{CCIO}. When you select the FPP scheme as a default in the Quartus II software, these I/O pins are tri-stated in user mode. To change this default option in the Quartus II software, select the **Dual-Purpose Pins** tab of the **Device and Pin Options** dialog box.

The configuration clock (DCLK) speed must be below the specified frequency to ensure correct configuration. No maximum DCLK period exists, which means you can pause configuration by halting DCLK for an indefinite amount of time.

If you are using the Stratix III decompression feature, design security feature, or both and need to stop DCLK, it can only be stopped three clock cycles after the last data byte was latched into the Stratix III device. If you are using the Stratix III device without decompression or design security feature, the DCLK can only be stopped two clock cycles after the last data byte was latched into the Stratix III device.

By stopping DCLK, the configuration circuit allows enough clock cycles to process the last byte of latched configuration data. When the clock restarts, the MAX II device must provide data on the DATA[7..0] pins prior to sending the first DCLK rising edge.

If an error occurs during configuration, the device drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the MAX II device that there is an error. If the **Auto-restart configuration after error** option (available in the Quartus II software on the **General** tab of the **Device and Pin Options** dialog box) is turned on, the device releases nSTATUS after a reset time-out period (maximum of 100 µs). After nSTATUS is released and pulled high by a pull-up resistor, the MAX II device can try to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

If you have enabled the **Auto-restart configuration after error** option, the nSTATUS pin transitions from high to low and back again to high when a configuration error is detected. This appears as a low pulse at the nSTATUS pin with a minimum pulse width of 10 µs to a maximum pulse width of 500 µs, as defined in the t_{STATUS} specification.

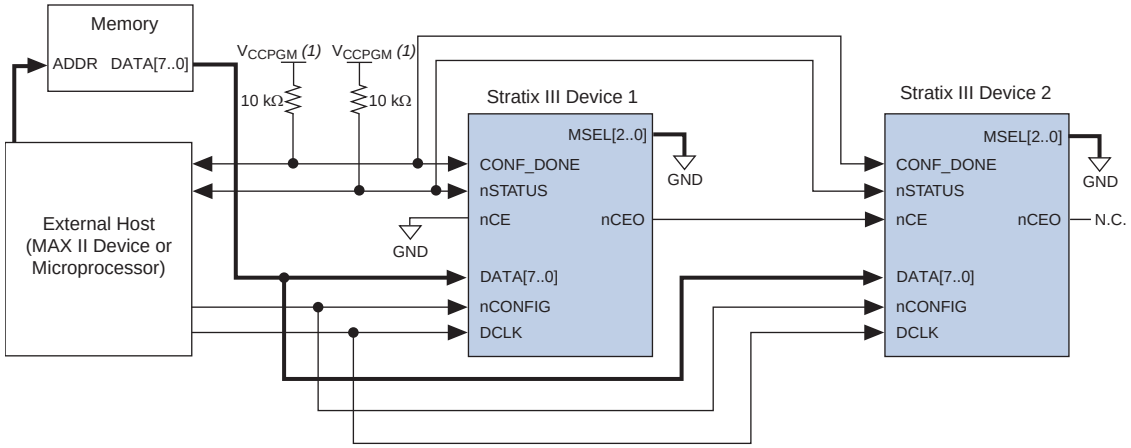
The MAX II device can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The MAX II device must monitor the CONF_DONE pin to detect errors and determine when programming completes. If all configuration data is sent, but the CONF_DONE or INIT_DONE signals have not gone high, the MAX II device will reconfigure the target device.

If you use the optional CLKUSR pin and the nCONFIG is pulled low to restart configuration during device initialization, you must ensure CLKUSR continues toggling during the time nSTATUS is low (maximum of 100 µs).

When the device is in user mode, transitioning the nCONFIG pin low to high initiates a reconfiguration. The nCONFIG pin should be low for at least 2 µs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. After nCONFIG returns to a logic high level and nSTATUS is released by the device, reconfiguration begins.

Figure 11–4 shows how to configure multiple devices using a MAX II device. This circuit is similar to the FPP configuration circuit for a single device, except the Stratix III devices are cascaded for multi-device configuration.

Figure 11–4. Multi-Device FPP Configuration Using an External Host



Note to Figure 11–4:

(1) Connect the resistor to a supply that provides an acceptable input signal for all Stratix III devices on the chain. V_{CCPGM} should be high enough to meet the V_{IH} specification of the I/O on the external host. It is recommended to power up all configuration system's I/O with V_{CCPGM}.

In a multi-device FPP configuration, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration within one clock cycle; therefore, the transfer of data destinations is transparent to the MAX II device. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. The configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

All nSTATUS and CONF_DONE pins are tied together. If any device detects an error, configuration stops for the entire chain and you must reconfigure the entire chain. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

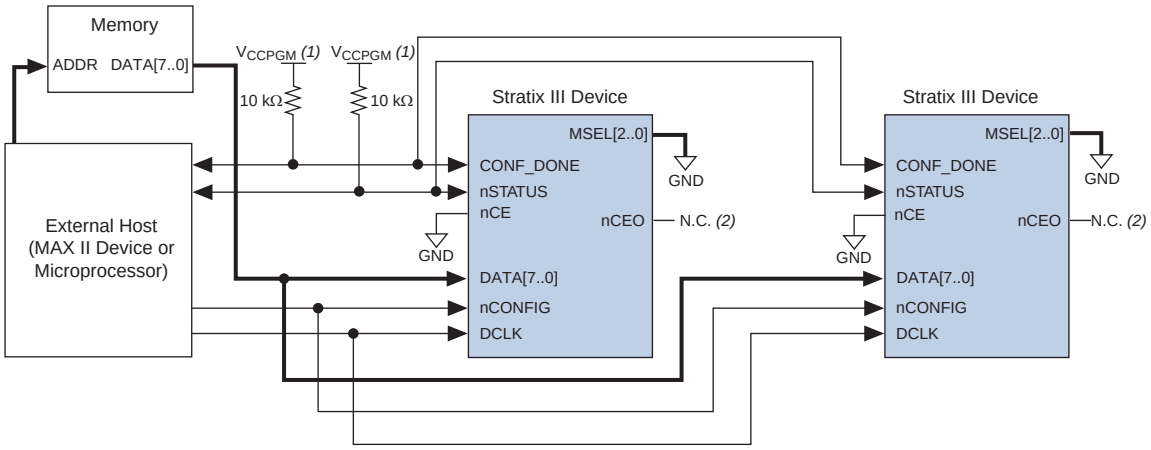
If the **Auto-restart configuration after error** option is turned on, the devices release their nSTATUS pins after a reset time-out period (maximum of 100 µs). After all nSTATUS pins are released and pulled high, the MAX II device tries to reconfigure the chain without pulsing nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

If you have enabled the **Auto-restart configuration after error** option, the nSTATUS pin transitions from high to low and back again to high when a configuration error is detected. This appears as a low pulse at the nSTATUS pin with a minimum pulse width of 10 µs to a maximum pulse width of 500 µs, as defined in the t_{STATUS} specification.

In a multi-device FPP configuration chain, all Stratix III devices in the chain must either enable or disable the decompression feature, design security feature, or both. You cannot selectively enable the decompression feature, design security feature, or both for each device in the chain because of the DATA and DCLK relationship. If the chain contains devices that do not support design security, you should use a serial configuration scheme.

If a system has multiple devices that contain the same configuration data, tie all device nCE inputs to GND, and leave nCEO pins floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. Configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices start and complete configuration at the same time. Figure 11–5 shows a multi-device FPP configuration when both Stratix III devices are receiving the same configuration data.

Figure 11–5. Multiple-Device FPP Configuration Using an External Host When Both Devices Receive the Same Data



Notes to Figure 11–5:

(1) Connect the resistor to a supply that provides an acceptable input signal for all Stratix III devices on the chain. V_{CCPGM} should be high enough to meet the V_{IH} specification of the I/O on the external host. It is recommended to power up all configuration system's I/O with V_{CCPGM}.

(2) The nCEO pins of both devices are left unconnected when configuring the same configuration data into multiple devices.

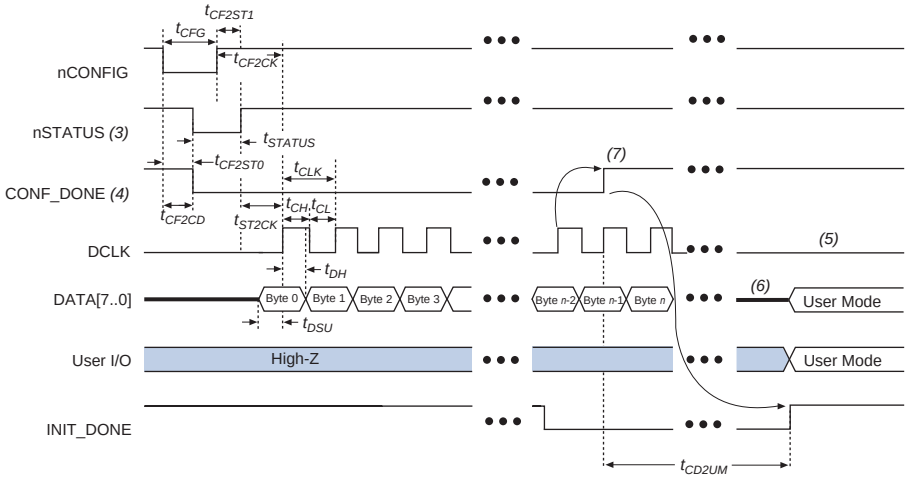
You can use a single configuration chain to configure Stratix III devices with other Altera devices that support FPP configuration, such as other types of Stratix devices. To ensure that all devices in the chain complete configuration at the same time, or that an error flagged by one device initiates reconfiguration in all devices, tie all of the device CONF_DONE and nSTATUS pins together.

For more information about configuring multiple Altera devices in the same configuration chain, refer to *Configuring Mixed Altera FPGA Chains* in the *Configuration Handbook*.

FPP Configuration Timing

Figure 11–6 shows the timing waveform for FPP configuration using a MAX II device as an external host. This waveform shows the timing when the decompression and the design security feature are not enabled.

Figure 11–6. FPP Configuration Timing Waveform *(Note 1)*, *(2)*



Notes to Figure 11–6:

(1) You should use this timing waveform when the decompression and design security features are not used.

(2) The beginning of this waveform shows the device in user mode. In user mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.

(3) Upon power-up, the Stratix III device holds nSTATUS low for the time of the POR delay.

(4) Upon power-up, before and during configuration, CONF_DONE is low.

(5) You should not leave DCLK floating after configuration. You should drive it high or low, whichever is more convenient.

(6) DATA[7..0] are available as user I/O pins after configuration. The state of these pins depends on the dual-purpose pin settings.

(7) Two DCLK falling edges are required after CONF_DONE goes high to begin the initialization of the device.

Table 11–5 defines the timing parameters for Stratix III devices for FPP configuration when the decompression and the design security features are not enabled.

Table 11–5. FPP Timing Parameters for Stratix III Devices *(Note 1)* (Part 1 of 2)

Symbol	Parameter	Minimum	Maximum	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low	—	800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low	—	800	ns
t_{CFG}	nCONFIG low pulse width	2	—	µs
t_{STATUS}	nSTATUS low pulse width	10	100 *(2)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high	—	100 *(2)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	100	—	µs
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2	—	µs
t_{DSU}	Data setup time before rising edge on DCLK	5	—	ns
t_{DH}	Data hold time after rising edge on DCLK	0	—	ns
t_{CH}	DCLK high time	4	—	ns
t_{CL}	DCLK low time	4	—	ns

Table 11–5. FPP Timing Parameters for Stratix III Devices *(Note 1)* (Part 2 of 2)

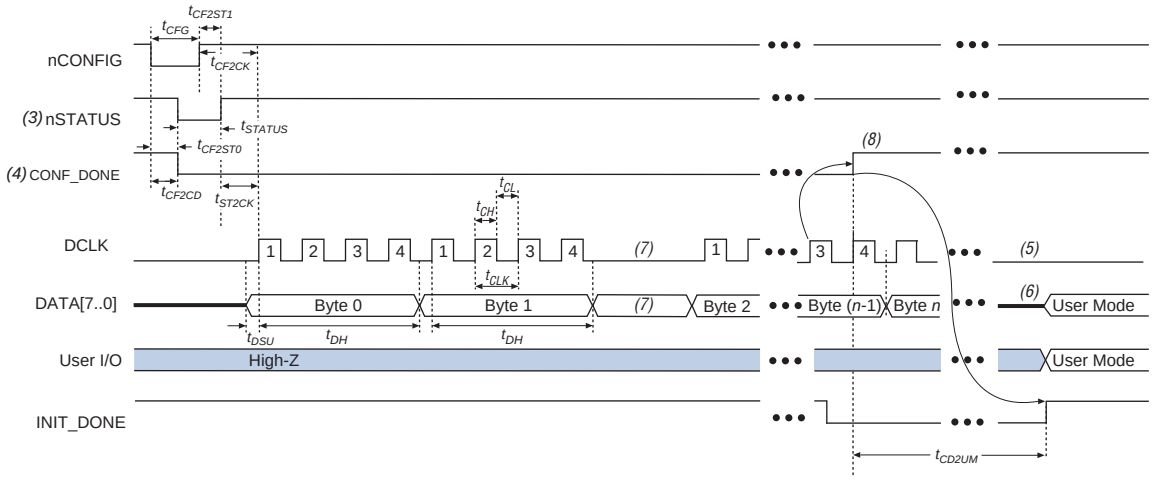
Symbol	Parameter	Minimum	Maximum	Units
t_{CLK}	DCLK period	10	—	ns
f_{MAX}	DCLK frequency	—	100	MHz
t_R	Input rise time	—	40	ns
t	Input fall time	—	40	ns
t_{CD2UM}	CONF_DONE high to user mode *(3)*	20	100	µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period	—	—
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	tCD2CU + (4,436 × CLKUSR period)	—	—

Notes to Table 11–5:

(1) Use these timing parameters when the decompression and design security features are not used.
(2) This value is obtainable if you do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.
(3) The minimum and maximum numbers apply only if you chose the internal oscillator as the clock source for starting up the device.

Figure 11–7 shows the timing waveform for FPP configuration when using a MAX II device as an external host. This waveform shows the timing when the decompression feature, design security feature, or both are enabled.

Figure 11–7. FPP Configuration Timing Waveform with Decompression or Design Security Feature Enabled *(Note 1)*, *(2)*



Notes to Figure 11–7:

(1) Use this timing waveform when the decompression feature, design security feature, or both are used.
(2) The beginning of this waveform shows the device in user mode. In user mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.
(3) Upon power-up, the Stratix III device holds nSTATUS low for the time of the POR delay.
(4) Upon power-up, before and during configuration, CONF_DONE is low.
(5) Do not leave DCLK floating after configuration. Drive it high or low, whichever is more convenient.
(6) DATA[7..0] are available as user I/O pins after configuration. The state of these pins depends on the dual-purpose pin settings.
(7) If needed, pause DCLK by holding it low. When DCLK restarts, the external host must provide data on the DATA[7..0] pins prior to sending the first DCLK rising edge.
(8) Two DCLK falling edges are required after CONF_DONE goes high to begin the initialization of the device.

Table 11–6 defines the timing parameters for Stratix III devices for FPP configuration when the decompression feature, design security feature, or both are enabled.

Table 11–6. FPP Timing Parameters for Stratix III Devices with Decompression or Design Security Feature Enabled *(Note 1)*

Symbol	Parameter	Minimum	Maximum	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low	—	800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low	—	800	ns
t_{CFG}	nCONFIG low pulse width	2	—	µs
t_{STATUS}	nSTATUS low pulse width	10	100 *(2)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high	—	100 *(2)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	100	—	µs
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2	—	µs
t_{DSU}	Data setup time before rising edge on DCLK	5	—	ns
t_{DH}	Data hold time after rising edge on DCLK	30	—	ns
t_{CH}	DCLK high time	4	—	ns
t_{CL}	DCLK low time	4	—	ns
t_{CLK}	DCLK period	10	—	ns
f_{MAX}	DCLK frequency	—	100	MHz
t_{DATA}	Data rate	—	200	Mbps
t_R	Input rise time	—	40	ns
t	Input fall time	—	40	ns
t_{CD2UM}	CONF_DONE high to user mode *(3)*	20	100	µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period	—	—
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	t_{CD2CU} + (4,436 × CLKUSR period)	—	—

Notes to Table 11–6:

(1) Use these timing parameters when the decompression and design security features are used.
(2) This value is obtainable if you do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.
(3) The minimum and maximum numbers apply only if you chose the internal oscillator as the clock source for starting up the device.

Device configuration options and how to create configuration files are discussed further in the *Device Configuration Options* and *Configuration File Formats* chapters in volume 2 of the *Configuration Handbook*.

FPP Configuration Using a Microprocessor

In this configuration scheme, a microprocessor can control the transfer of configuration data from a storage device, such as flash memory, to the target Stratix III device.

All information in "FPP Configuration Using a MAX II Device as an External Host" on page 11–8 is also applicable when using a microprocessor as an external host. Refer to this section for all configuration and timing information.

Fast Active Serial Configuration (Serial Configuration Devices)

In the fast AS configuration scheme, Stratix III devices are configured using a serial configuration device. These configuration devices are low-cost devices with non-volatile memory that feature a simple four-pin interface and a small form factor. These features make serial configuration devices an ideal low-cost configuration solution.

For more information about serial configuration devices, refer to the *Serial Configuration Devices Data Sheet* in the *Configuration Handbook*.

Serial configuration devices provide a serial interface to access configuration data. During device configuration, Stratix III devices read configuration data through the serial interface, decompress data if necessary, and configure their SRAM cells. This scheme is referred to as the AS configuration scheme, because the Stratix III device controls the configuration interface. This scheme contrasts with the PS configuration scheme, where the configuration device controls the interface.

The Stratix III decompression and design security features are fully available when configuring your Stratix III device using fast AS mode.

Table 11–7 lists the MSEL pin settings for the AS configuration scheme.

Table 11–7. Stratix III MSEL Pin Settings for AS Configuration Schemes *(Note 1)*

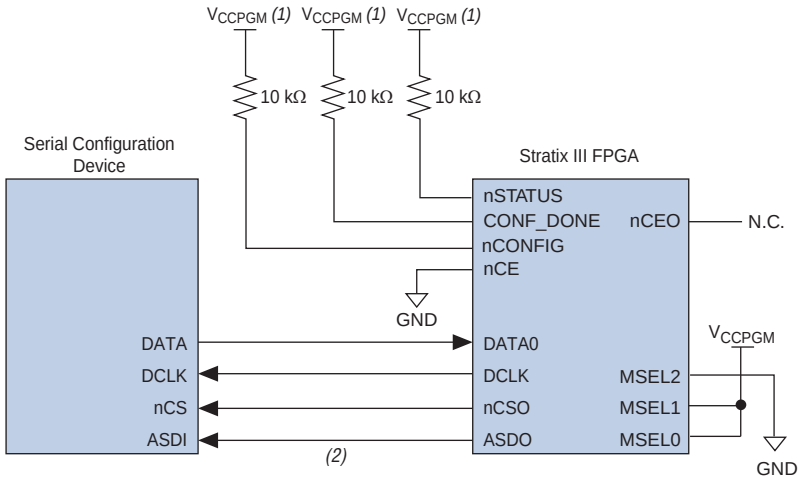
Configuration Scheme	MSEL2	MSEL1	MSEL0
Fast AS (40 MHz)	0	1	1
Remote system upgrade fast AS (40 MHz)	0	1	1

Note to Table 11–7:

(1) Use EPCS16, EPCS64, or EPCS128 devices.

Serial configuration devices have a four-pin interface: serial clock input (DCLK), serial data output (DATA), AS data input (ASDI), and an active-low chip select (nCS). This four-pin interface connects to Stratix III device pins, as shown in Figure 11–8.

Figure 11–8. Single Device Fast AS Configuration



Notes to Figure 11–8:

(1) Connect the pull-up resistors to V_{CCPGM} at 3.3-V supply.

(2) Stratix III devices use the ASDO-to-ASDI path to control the configuration device.

Upon power-up, the Stratix III devices go through a POR. The POR delay is dependent on the PORSEL pin setting. When PORSEL is driven low, the POR time is approximately 100 ms. If PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device will reset, hold nSTATUS and CONF_DONE low, and tri-state all user I/O pins. After the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. After POR, the Stratix III device releases nSTATUS, which is pulled high by an external 10-kΩ pull-up resistor and enters configuration mode.

To begin configuration, power the V_{CC}, V_{CCIO}, V_{CCPGM}, and V_{CCPD} voltages (for the banks where the configuration and JTAG pins reside) to the appropriate voltage levels.

The serial clock (DCLK) generated by the Stratix III device controls the entire configuration cycle and provides the timing for the serial interface. Stratix III devices use an internal oscillator to generate DCLK. Using the MSEL[] pins, you can select to use a 40 MHz oscillator.

In fast AS configuration schemes, Stratix III devices drive out control signals on the falling edge of DCLK. The serial configuration device responds to the instructions by driving out configuration data on the falling edge of DCLK. Then the data is latched into the Stratix III device on the following falling edge of DCLK.

In configuration mode, Stratix III devices enable the serial configuration device by driving the nCSO output pin low, which connects to the chip select (nCS) pin of the configuration device. The Stratix III device uses the serial clock (DCLK) and serial data output (ASDO) pins to send operation commands, read address signals, or both to the serial configuration device. The configuration device provides data on its serial data output (DATA) pin, which connects to the DATA0 input of the Stratix III devices.

After all configuration bits are received by the Stratix III device, it releases the open-drain CONF_DONE pin, which is pulled high by an external 10-kΩ resistor. Initialization begins only after the CONF_DONE signal reaches a logic high level. All AS configuration pins (DATA0, DCLK, nCSO, and ASDO) have weak internal pull-up resistors that are always active. After configuration, these pins are set as input tri-stated and are driven high by the weak internal pull-up resistors. The CONF_DONE pin must have an external 10-kΩ pull-up resistor for the device to initialize.

In Stratix III devices, the initialization clock source is either the 10 MHz (typical) internal oscillator (separate from the active serial internal oscillator) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If you use the internal oscillator, the Stratix III device has enough clock cycles for proper initialization. You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. You can turn on the **Enable user-supplied start-up clock** (**CLKUSR**) option in the Quartus II software on the **General** tab of the **Device and Pin Options** dialog box. When you enable the **user supplied start-up clock** option, the CLKUSR pin is the initialization clock source. Supplying a clock on CLKUSR does not affect the configuration process. When all configuration data has been accepted and CONF_DONE goes high, CLKUSR is enabled after 600 ns. After this time period elapses, Stratix III devices require 4,436 clock cycles to initialize properly and enter user mode. Stratix III devices support a CLKUSR f_{MAX} of 100 MHz.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The **Enable INIT_DONE Output** option is available in the Quartus II software on the **General** tab of the **Device and Pin Options** dialog box. If you use the INIT_DONE pin, it will be high due to an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. When the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. This low-to-high transition signals that the device has entered user mode. When initialization is complete, the device enters user mode. In user mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

If an error occurs during configuration, Stratix III devices assert the nSTATUS signal low, indicating a data frame error, and the CONF_DONE signal stays low. If the **Auto-restart configuration after error** option (available in the Quartus II software on the **General** tab of the **Device and Pin Options** dialog box) is turned on, the Stratix III device resets the configuration device by pulsing nCSO, releases nSTATUS after a reset time-out period (maximum of 100 µs), and retries configuration. If this option is turned off, the system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 2 µs to restart configuration.

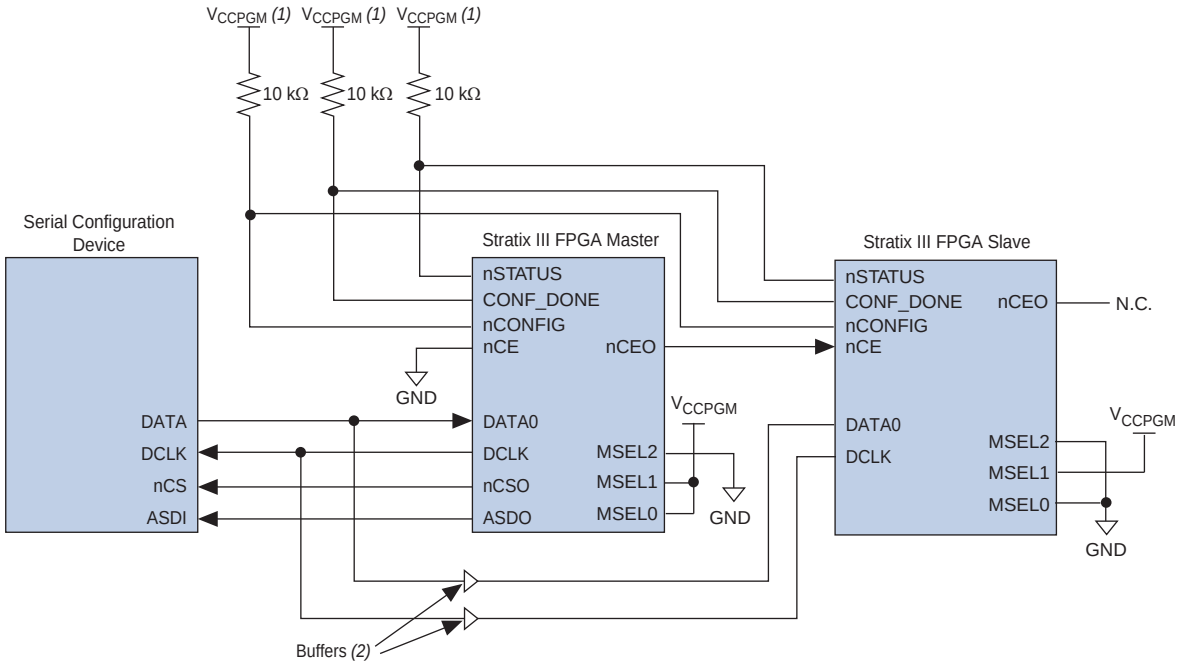
If you have enabled the **Auto-restart configuration after error** option, the nSTATUS pin transitions from high to low and back again to high when a configuration error is detected. This appears as a low pulse at the nSTATUS pin with a minimum pulse width of 10 µs to a maximum pulse width of 500 µs, as defined in the t_{STATUS} specification.

When the Stratix III device is in user mode, you can initiate reconfiguration by pulling the nCONFIG pin low. The nCONFIG pin should be low for at least 2 µs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. When nCONFIG returns to a logic high level and nSTATUS is released by the Stratix III device, reconfiguration begins.

You can configure multiple Stratix III devices using a single serial configuration device. You can cascade multiple Stratix III devices using the chip-enable (nCE) and chip-enable-out (nCEO) pins. The first device in the chain must have its nCE pin connected to ground. You must connect its nCEO pin to the nCE pin of the next device in the chain. When the first device captures all of its configuration data from the bitstream, it drives the nCEO pin low, enabling the next device in the chain. You must leave the nCEO pin of the last device unconnected. The nCONFIG, nSTATUS, CONF_DONE, DCLK, and DATA0 pins of each device in the chain are connected (refer to Figure 11–9).

This first Stratix III device in the chain is the configuration master and controls configuration of the entire chain. You must connect its MSEL pins to select the AS configuration scheme. The remaining Stratix III devices are configuration slaves. You must connect their MSEL pins to select the PS configuration scheme. Any other Altera device that supports PS configuration can also be part of the chain as a configuration slave. Figure 11–9 shows the pin connections for this setup.

Figure 11–9. Multi-Device Fast AS Configuration



Notes to Figure 11–9:

(1) Connect the pull-up resistors to V_{CCPGM} at 3.3-V supply.

(2) Connect the repeater buffers between the Stratix III master and slave device(s) for DATA[0] and DCLK. This prevents any potential signal integrity and clock skew problems.

As shown in Figure 11–9, the nSTATUS and CONF_DONE pins on all target devices are connected with external pull-up resistors. These pins are open-drain bi-directional pins on the devices. When the first device asserts nCEO (after receiving all of its configuration data), it releases its CONF_DONE pin. The subsequent devices in the chain keep this shared CONF_DONE line low until they have received their configuration data. When all target devices in the chain have received their configuration data and released CONF_DONE, the pull-up resistor drives a high level on this line and all devices simultaneously enter initialization mode.

If an error occurs at any point during configuration, the nSTATUS line is driven low by the failing device. If you enable the **Auto-restart configuration after error** option, reconfiguration of the entire chain begins after a reset time-out period (maximum of 100 µs). If the **Auto-restart configuration after error** option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low to restart configuration. The external system can pulse nCONFIG if it is under system control rather than tied to V_{CCPGM}.

If you have enabled the **Auto-restart configuration after error** option, the nSTATUS pin transitions from high to low and back again to high when a configuration error is detected. This appears as a low pulse at the nSTATUS pin with a minimum pulse width of 10 µs to a maximum pulse width of 500 µs, as defined in the t_{STATUS} specification.

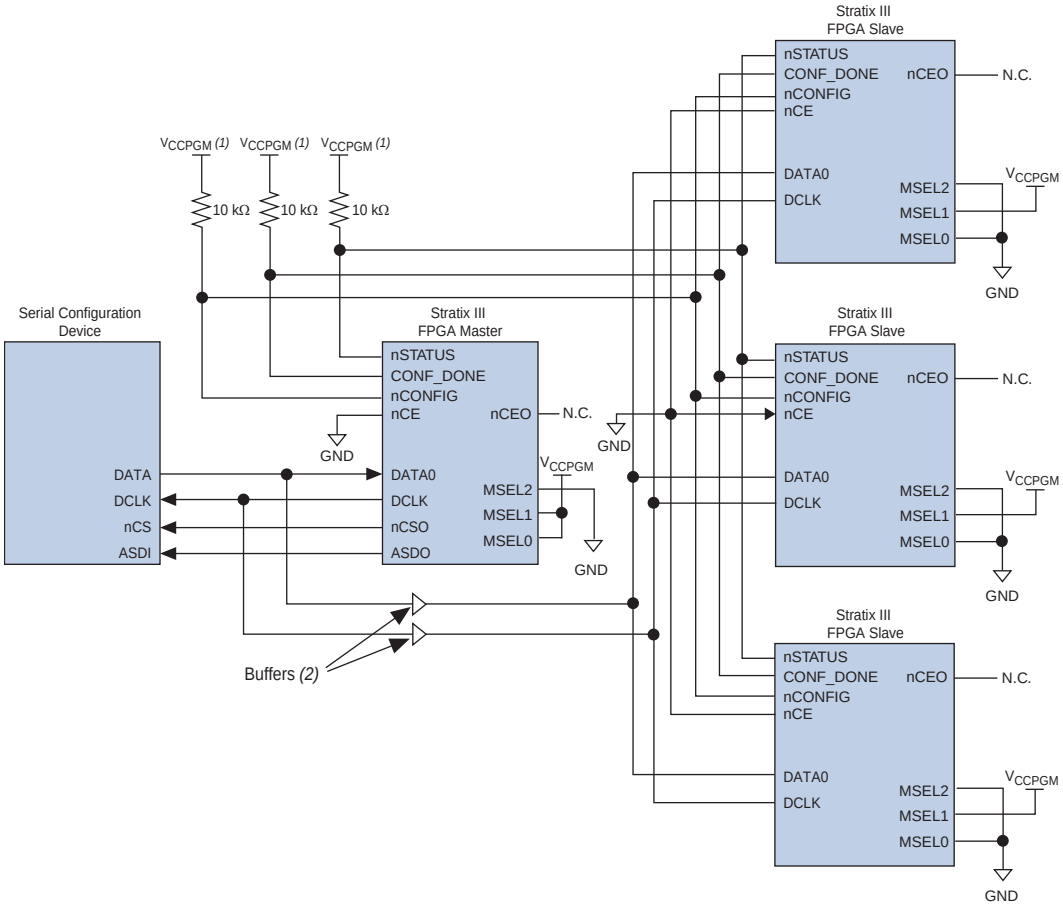
While you can cascade Stratix III devices, you cannot cascade or chain together serial configuration devices.

If the configuration bitstream size exceeds the capacity of a serial configuration device, you must select a larger configuration device, enable the compression feature, or both. When configuring multiple devices, the size of the bitstream is the sum of the individual devices' configuration bitstreams.

A system may have multiple devices that contain the same configuration data. In active serial chains, you can implement this by storing one copy of the SOF in the serial configuration device. The same copy of the SOF configures the master Stratix III device and all remaining slave devices concurrently. All Stratix III devices must be the same density and package. The master device is set up in active serial mode and the slave devices are set up in passive serial mode.

To configure four identical Stratix III devices with the same SOF, you could set up the chain similar to the example shown in Figure 11–10. The first device is the master device, and its MSEL pins should be set to select AS configuration. The other three slave devices are set up for concurrent configuration, and their MSEL pins should be set to select PS configuration. The nCE input pins from the master and slave are connected to GND, and the DATA and DCLK pins connect in parallel to all four devices. During the configuration cycle, the master device reads its configuration data from the serial configuration device and transmits the second copy of the configuration data to all three slave devices, configuring all of them simultaneously.

Figure 11–10. Multi-Device Fast AS Configuration When the Devices Receive the Same Data Using a Single SOF



Notes to Figure 11–10:

(1) Connect the pull-up resistors to V_{CCPGM} at 3.3-V supply.

(2) Connect the repeater buffers between the Stratix III master and slave device(s) for `DATA[0]` and `DCLK`. This prevents any potential signal integrity and clock skew problems.

Estimating Active Serial Configuration Time

Active serial configuration time is dominated by the time it takes to transfer data from the serial configuration device to the Stratix III device. This serial interface is clocked by the Stratix III `DCLK` output (generated from an internal oscillator). Because the Stratix III device only supports fast AS configuration, the `DCLK` frequency needs to be set to 40 MHz (25 ns). The minimum configuration time estimate for an EP3SL50 device (15 MBits of uncompressed data) is shown in Equation 11–1 and Example 11–1.

Equation 11–1.

$$\text{RBF Size} \times (\text{minimum DCLK period} / 1 \text{ bit per DCLK cycle}) = \text{estimated minimum configuration time}$$

Example 11–1.

$$15 \text{ Mbits} \times (25 \text{ ns} / 1 \text{ bit}) = 375 \text{ ms}$$

Enabling compression reduces the amount of configuration data that is transmitted to the Stratix III device, which also reduces configuration time. On average, compression reduces configuration time, depending on the design.

Figure 11–11 shows the timing waveform for AS configuration.

Figure 11–11. Fast AS Configuration Timing

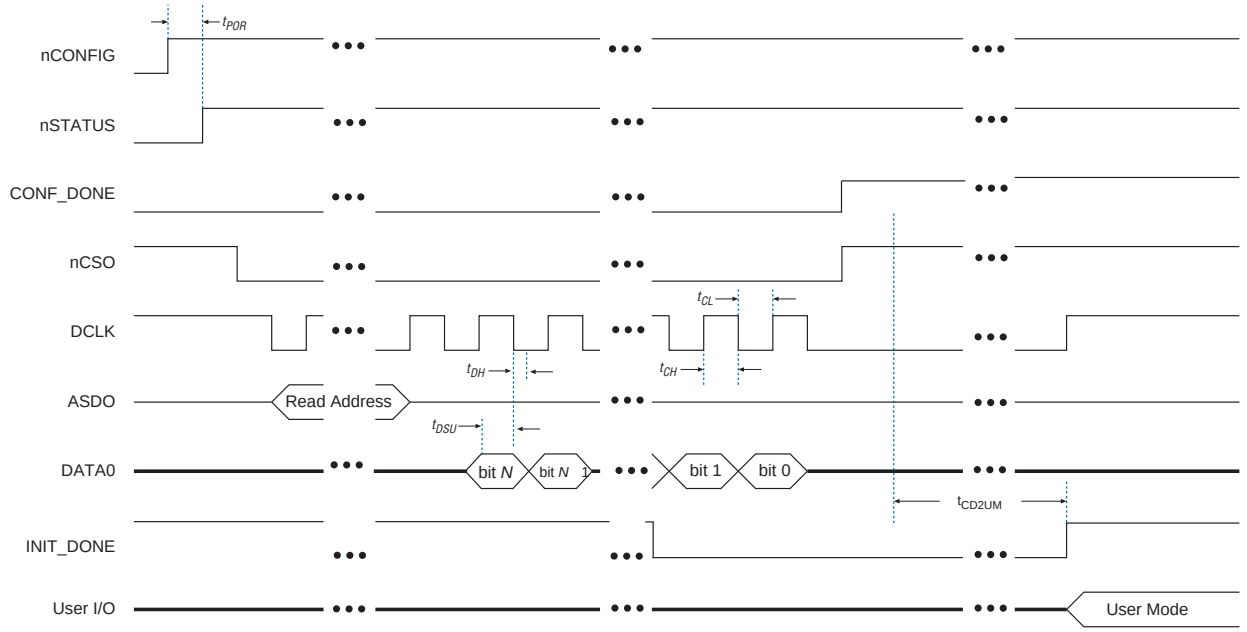


Table 11–8 defines the timing parameters for Stratix III devices for fast AS configuration.

Table 11–8. Fast AS Timing Parameters for Stratix III Devices

Symbol	Parameter	Min	Typ	Max	Units
f_{CLK}	DCLK frequency from Stratix III	15	25	40	MHz
t_{CF2ST1}	nCONFIG high to nSTATUS high	—	—	100	µs
t_{DSU}	Data setup time before falling edge on DCLK	7	—	—	ns
t_{DH}	Data hold time after falling edge on DCLK	0	—	—	ns
t_{CH}	DCLK high time	10	—	—	ns
t_{CL}	DCLK low time	10	—	—	ns
t_{CD2UM}	CONF_DONE high to user mode	20	—	100	µs

Programming Serial Configuration Devices

Serial configuration devices are non-volatile, flash-memory-based devices. You can program these devices in-system using the USB-Blaster™ or ByteBlaster™ II download cable. Alternatively, you can program them using the Altera programming unit (APU), supported third-party programmers, or a microprocessor with the SRunner software driver.

You can perform in-system programming of serial configuration devices by using the conventional AS programming interface or JTAG interface solution.

Because serial configuration devices do not support the JTAG interface, the conventional method to program them is by using the AS programming interface. The configuration data used to program serial configuration devices is downloaded by using the programming hardware.

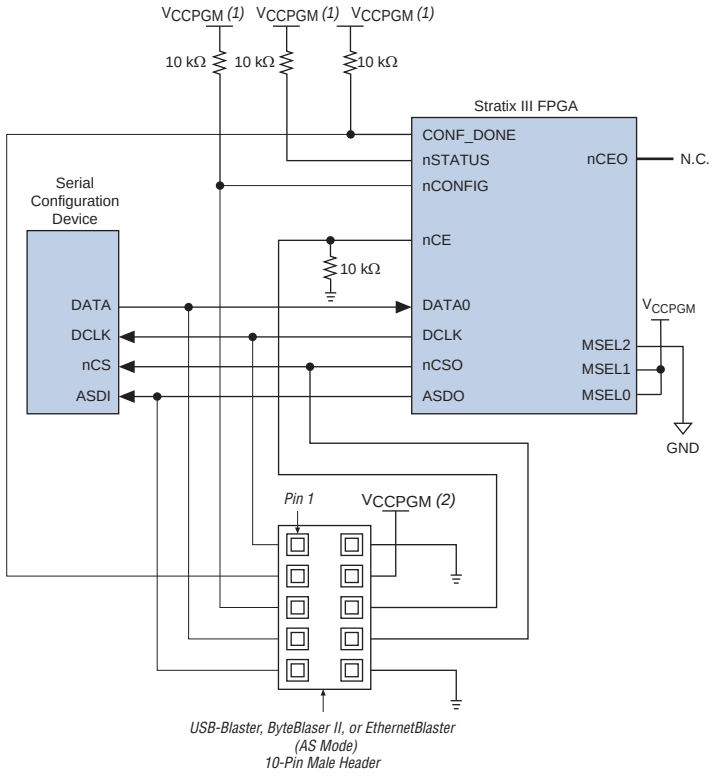
During in-system programming, the download cable disables device access to the AS interface by driving the `nCE` pin high. Stratix III devices are also held in reset by a low level on `nCONFIG`. After programming is complete, the download cable releases `nCE` and `nCONFIG`, allowing the pull-down and pull-up resistors to drive GND and V_{CCPGM}, respectively. Figure 11–12 shows the download cable connections to the serial configuration device.

Altera has developed the Serial FlashLoader (SFL), which is an in-system programming solution for serial configuration devices using the JTAG interface. This solution requires the Stratix III device to be a bridge between the JTAG interface and the serial configuration device.

For more information about the SFL, refer to *AN 370: Using the Serial FlashLoader with the Quartus II Software*.

For more information about the USB-Blaster download cable, refer to the *USB-Blaster Download Cable User Guide*. For more information about the ByteBlaster II cable, refer to the *ByteBlaster II Download Cable User Guide*.

Figure 11–12. In-System Programming of Serial Configuration Devices



Notes to Figure 11–12:

(1) Connect the pull-up resistors to V_{CCPGM} at 3.3-V supply.

(2) Power up the USB-Blaster, ByteBlaster II, or EthernetBlaster cable's $V_{CC(TRGT)}$ with V_{CCPGM}.

You can program serial configuration devices with the Quartus II software using the Altera programming hardware and the appropriate configuration device programming adapter.

In production environments, you can program serial configuration devices using multiple methods. You can use Altera programming hardware or other third-party programming hardware to program blank serial configuration devices before they are mounted onto PCBs. Alternatively, you can use an on-board microprocessor to program the serial configuration device in-system using C-based software drivers provided by Altera.

You can program a serial configuration device in-system by an external microprocessor using SRunner. SRunner is a software driver developed for embedded serial configuration device programming that can be easily customized to fit in different embedded systems. SRunner is able to read a raw programming data (.**rpd**) file and write to the serial configuration devices. The serial configuration device programming time using SRunner is comparable to the programming time with the Quartus II software.

For more information about SRunner, refer to *AN 418: SRunner: An Embedded Solution for EPCS Programming* and the source code on the Altera website at www.altera.com.

For more information about programming serial configuration devices and fast AS Configuration Timing, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in the *Configuration Handbook*.

Passive Serial Configuration

You can program PS configuration of Stratix III devices using an intelligent host, such as a MAX II device or microprocessor with flash memory, or a download cable. In the PS scheme, an external host (a MAX II device, embedded processor, or host PC) controls configuration. Configuration data is clocked into the target Stratix III device by using the DATA0 pin at each rising edge of DCLK.

The Stratix III decompression and design security features are fully available when configuring your Stratix III device using PS mode.

Table 11–9 lists the MSEL pin settings when using the PS configuration scheme.

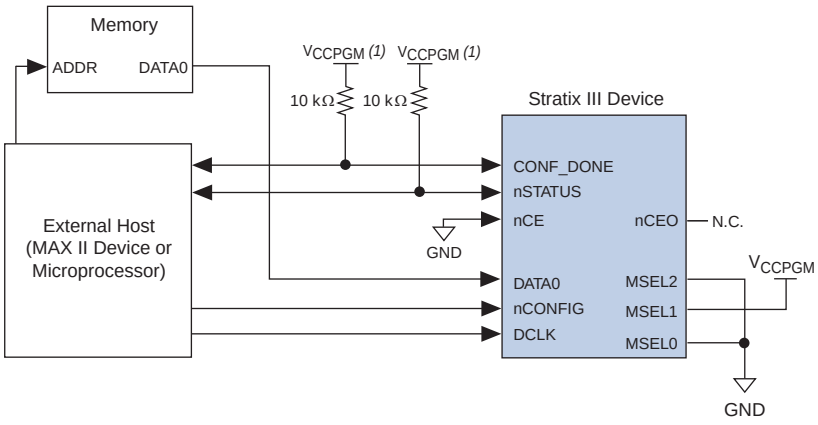
Table 11–9. Stratix III MSEL Pin Settings for PS Configuration Scheme

Configuration Scheme	MSEL2	MSEL1	MSEL0
PS	0	1	0

PS Configuration Using a MAX II Device as an External Host

In this configuration scheme, you can use a MAX II device as an intelligent host that controls the transfer of configuration data from a storage device, such as flash memory, to the target Stratix III device. You can store configuration data in **.rbf**, **.hex**, or **.ttf** format. Figure 11–13 shows the configuration interface connections between a Stratix III device and a MAX II device for single device configuration.

Figure 11–13. Single Device PS Configuration Using an External Host



Note to Figure 11–13:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Stratix III device. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the external host. It is recommended to power up all configuration systems' I/O with V_{CCPGM}.

Upon power-up, Stratix III devices go through a POR. The POR delay is dependent on the PORSEL pin setting. When PORSEL is driven low, the POR time is approximately 100 ms. When PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device resets, holds nSTATUS low, and tri-states all user I/O pins. When the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors that are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration, the MAX II device must generate a low-to-high transition on the nCONFIG pin.

V_{CC}, V_{CCIO}, V_{CCPGM}, and V_{CCPD} of the banks where the configuration and JTAG pins reside must be fully powered to the appropriate voltage levels to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. When nSTATUS is released, the device is ready to receive configuration data and the configuration stage begins. When nSTATUS is pulled high, the MAX II device should place the configuration data one bit at a time on the DATA0 pin. If you are using configuration data in **.rbf**, **.hex**, or **.ttf** format, you must send the least significant bit (LSB) of each data byte first. For example, if the RBF contains the byte sequence `02 1B EE 01 FA`, the serial bitstream you must transmit to the device is `0100-0000 1101-1000 0111-0111 1000-0000 0101-1111`.

The Stratix III device receives configuration data on the DATA0 pin and the clock is received on the DCLK pin. Data is latched into the device on the rising edge of DCLK. Data is continuously clocked into the target device until CONF_DONE goes high. After the device has received all configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by an external 10-kΩ pull-up resistor. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin. The CONF_DONE pin must have an external 10-kΩ pull-up resistor for the device to initialize.

In Stratix III devices, the initialization clock source is either the internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If you use the internal oscillator, the Stratix III device has enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device. Driving DCLK to the device after configuration is complete does not affect device operation.

You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. You can turn on the **Enable user-supplied start-up clock** (**CLKUSR**) option in the Quartus II software on the **General** tab of the **Device and Pin Options** dialog box. If you supply a clock on CLKUSR, it will not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, CLKUSR will be enabled after the time specified as t_{CD2CU}. After this time period elapses, Stratix III devices require 4,436 clock cycles to initialize properly and enter user mode. Stratix III devices support a CLKUSR f_{MAX} of 100 MHz.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The **Enable INIT_DONE Output** option is available in the Quartus II software on the **General** tab of the **Device and Pin Options** dialog box. If you use the INIT_DONE pin, it will be high due to an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. When the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low. When initialization is complete, the INIT_DONE pin will be released and pulled high. The MAX II device must be able to detect this low-to-high transition, which signals the device has entered user mode. When initialization is complete, the device enters user mode. In user mode, the user I/O pins will no longer have weak pull-up resistors and will function as assigned in your design.

To ensure DCLK and DATA0 are not left floating at the end of configuration, the MAX II device must drive them either high or low, whichever is convenient on your board. The DATA[0] pin is available as a user I/O pin after configuration. When you choose the PS scheme as a default in the Quartus II software, this I/O pin is tri-stated in user mode and should be driven by the MAX II device. To change this default option in the Quartus II software, click the **Dual-Purpose Pins** tab of the **Device and Pin Options** dialog box.

The configuration clock (DCLK) speed must be below the specified frequency to ensure correct configuration. No maximum DCLK period exists, which means you can pause configuration by halting DCLK for an indefinite amount of time.

If an error occurs during configuration, the device drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the MAX II device that there is an error. If the **Auto-restart configuration after error** option (available in the Quartus II software on the **General** tab of the **Device and Pin Options** dialog box) is turned on, the Stratix III device releases nSTATUS after a reset time-out period (maximum of 100 µs). After nSTATUS is released and pulled high by a pull-up resistor, the MAX II device can attempt to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

If you have enabled the **Auto-restart configuration after error** option, the nSTATUS pin transitions from high to low and back again to high when a configuration error is detected. This appears as a low pulse at the nSTATUS pin with a minimum pulse width of 10 µs to a maximum pulse width of 500 µs, as defined in the t_{STATUS} specification.

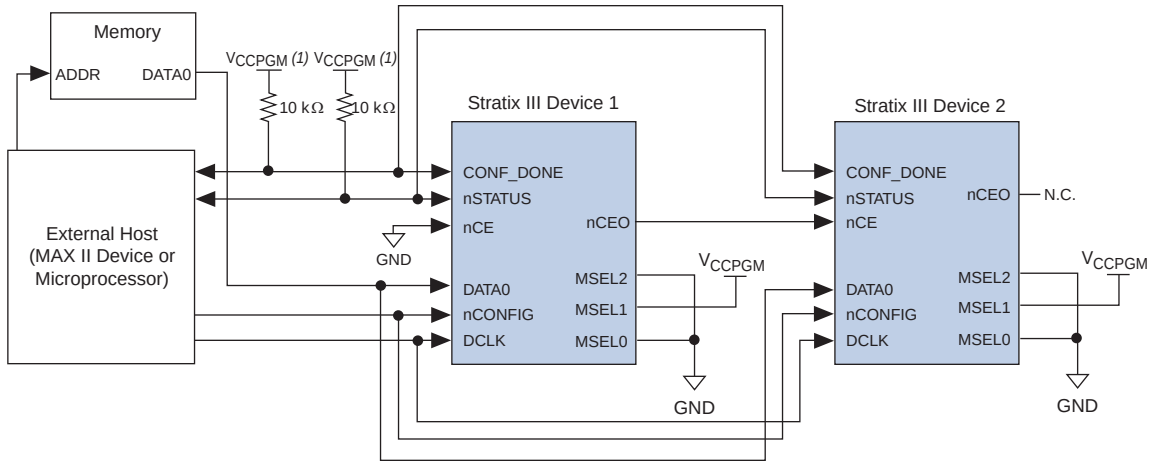
The MAX II device can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The CONF_DONE pin must be monitored by the MAX II device to detect errors and determine when programming completes. If all configuration data is sent, but CONF_DONE or INIT_DONE have not gone high, the MAX II device must reconfigure the target device.

If you use the optional CLKUSR pin and nCONFIG is pulled low to restart configuration during device initialization, you must ensure that CLKUSR continues toggling during the time nSTATUS is low (maximum of 100 µs).

When the device is in user mode, you can initiate a reconfiguration by transitioning the nCONFIG pin low-to-high. The nCONFIG pin must be low for at least 2 µs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. When nCONFIG returns to a logic high level and nSTATUS is released by the device, reconfiguration begins.

Figure 11–14 shows how to configure multiple devices using a MAX II device. This circuit is similar to the PS configuration circuit for a single device, except Stratix III devices are cascaded for multi-device configuration.

Figure 11–14. Multi-Device PS Configuration Using an External Host



Note to Figure 11–14:

(1) Connect the resistor to a supply that provides an acceptable input signal for all Stratix III devices on the chain. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the external host. It is recommended to power up all configuration systems' I/O with V_{CCPGM}.

In multi-device PS configuration, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration within one clock cycle. Therefore, the transfer of data destinations is transparent to the MAX II device. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

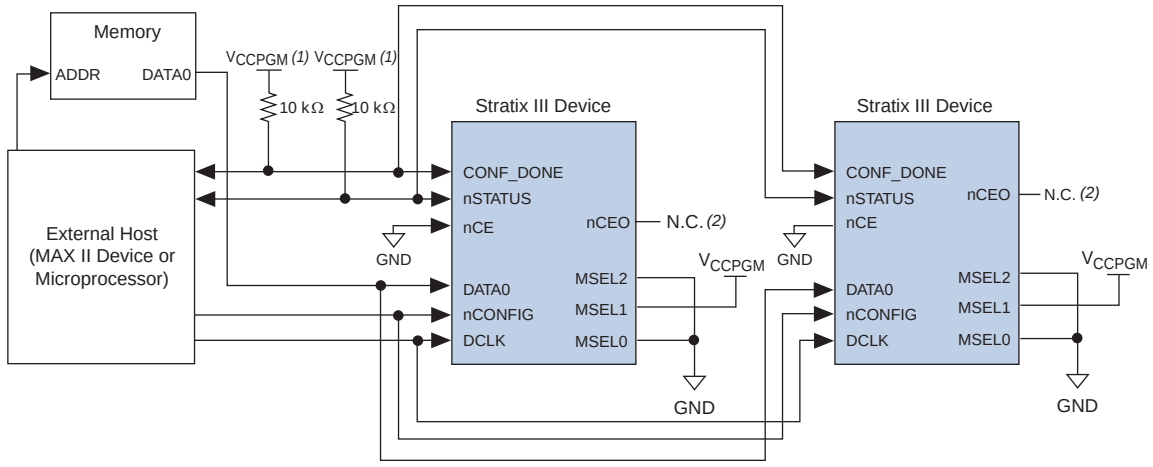
Since all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and you must reconfigure the entire chain. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

If the **Auto-restart configuration after error** option is turned on, the devices release their nSTATUS pins after a reset time-out period (maximum of 100 µs). After all nSTATUS pins are released and pulled high, the MAX II device can attempt to reconfigure the chain without needing to pulse nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

If you have enabled the **Auto-restart configuration after error** option, the nSTATUS pin transitions from high to low and back again to high when a configuration error is detected. This appears as a low pulse at the nSTATUS pin with a minimum pulse width of 10 µs to a maximum pulse width of 500 µs, as defined in the t_{STATUS} specification.

In your system, you can have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete configuration at the same time. Figure 11–15 shows multi-device PS configuration when both Stratix III devices are receiving the same configuration data.

Figure 11–15. Multiple-Device PS Configuration When Both Devices Receive the Same Data



Notes to Figure 11–15:

(1) Connect the resistor to a supply that provides an acceptable input signal for all Stratix III devices on the chain. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the external host. It is recommended to power up all configuration systems' I/O with V_{CCPGM}.

(2) The nCEO pins of both devices are left unconnected when configuring the same configuration data into multiple devices.

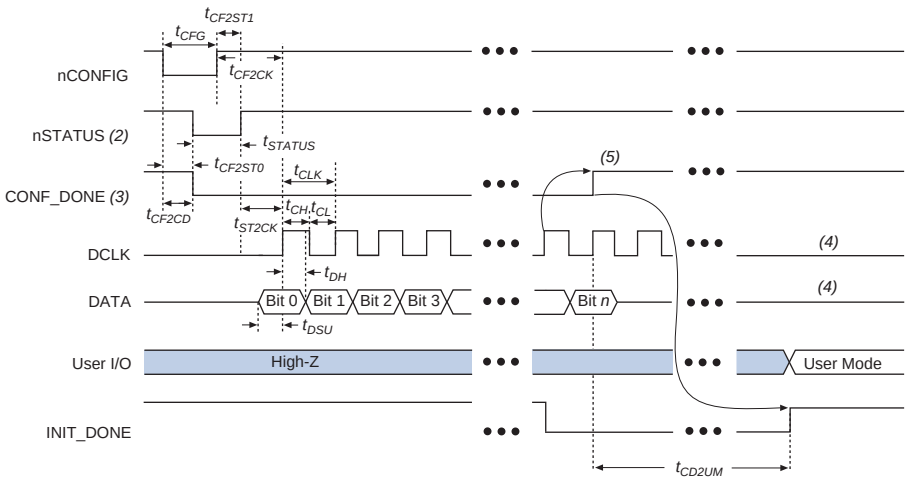
You can use a single configuration chain to configure Stratix III devices with other Altera devices. To ensure that all devices in the chain complete configuration at the same time, or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information about configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in the *Configuration Handbook*.

PS Configuration Timing

Figure 11–16 shows the timing waveform for PS configuration when using a MAX II device as an external host.

Figure 11–16. PS Configuration Timing Waveform *(Note 1)*



Notes to Figure 11–16:

(1) The beginning of this waveform shows the device in user-mode. In user-mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.

(2) Upon power-up, the Stratix III device holds nSTATUS low for the time of the POR delay.

(3) Upon power-up, before and during configuration, CONF_DONE is low.

(4) Do not leave DCLK floating after configuration. You should drive it high or low, whichever is more convenient. DATA[0] is available as a user I/O pin after configuration. The state of this pin depends on the dual-purpose pin settings.

(5) Two DCLK falling edges are required after CONF_DONE goes high to begin the initialization of the device.

Table 11–10 defines the timing parameters for Stratix III devices for PS configuration.

Table 11–10. PS Timing Parameters for Stratix III Devices (Part 1 of 2)

Symbol	Parameter	Minimum	Maximum	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low	—	800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low	—	800	ns
t_{CFG}	nCONFIG low pulse width	2	—	µs
t_{STATUS}	nSTATUS low pulse width	10	100 *(1)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high	—	100 *(1)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	100	—	µs
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2	—	µs
t_{DSU}	Data setup time before rising edge on DCLK	5	—	ns
t_{DH}	Data hold time after rising edge on DCLK	0	—	ns
t_{CH}	DCLK high time	4	—	ns
t_{CL}	DCLK low time	4	—	ns
t_{CLK}	DCLK period	10	—	ns
f_{MAX}	DCLK frequency	—	100	MHz
t_{R}	Input rise time	—	40	ns

Table 11–10. PS Timing Parameters for Stratix III Devices (Part 2 of 2)

Symbol	Parameter	Minimum	Maximum	Units
t	Input fall time	—	40	ns
t_{CD2UM}	CONF_DONE high to user mode *(2)*	20	100	µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period	—	—
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	tCD2CU + (4,436 × CLKUSR period)	—	—

Notes to Table 11–10:

(1) This value is applicable if you do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(2) The minimum and maximum numbers apply only if you choose the internal oscillator as the clock source for starting the device.

Device configuration options and how to create configuration files are discussed further in the *Device Configuration Options and Configuration File Formats* chapters in volume 2 of the *Configuration Handbook*.

PS Configuration Using a Microprocessor

In this PS configuration scheme, a microprocessor can control the transfer of configuration data from a storage device, such as flash memory, to the target Stratix III device.

You can do a PS configuration using MicroBlaster™ Passive Serial Software Driver. For more information, refer to *AN423: Configuring the MicroBlaster Passive Serial Software Driver*.

For all configuration and timing information, refer to "PS Configuration Using a MAX II Device as an External Host" on page 11–27. This section is also applicable when using a microprocessor as an external host.

PS Configuration Using a Download Cable

In this section, the generic term *download cable* includes the Altera USB-Blaster USB port download cable, MasterBlaster™ serial/USB communications cable, ByteBlaster II parallel port download cable, ByteBlasterMV™ parallel port download cable, and the EthernetBlaster download cable.

In PS configuration with a download cable, an intelligent host (such as a PC) transfers data from a storage device to the device by using the USB-Blaster, MasterBlaster, ByteBlaster II, EthernetBlaster, or ByteBlasterMV cable.

Upon power-up, the Stratix III devices go through a POR. The POR delay is dependent on the PORSEL pin setting. When PORSEL is driven low, the POR time is approximately 100 ms. If PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device will reset, hold nSTATUS low, and tri-state all user I/O pins. After the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

The configuration cycle consists of three stages: reset, configuration and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration in this scheme, the download cable generates a low-to-high transition on the nCONFIG pin.

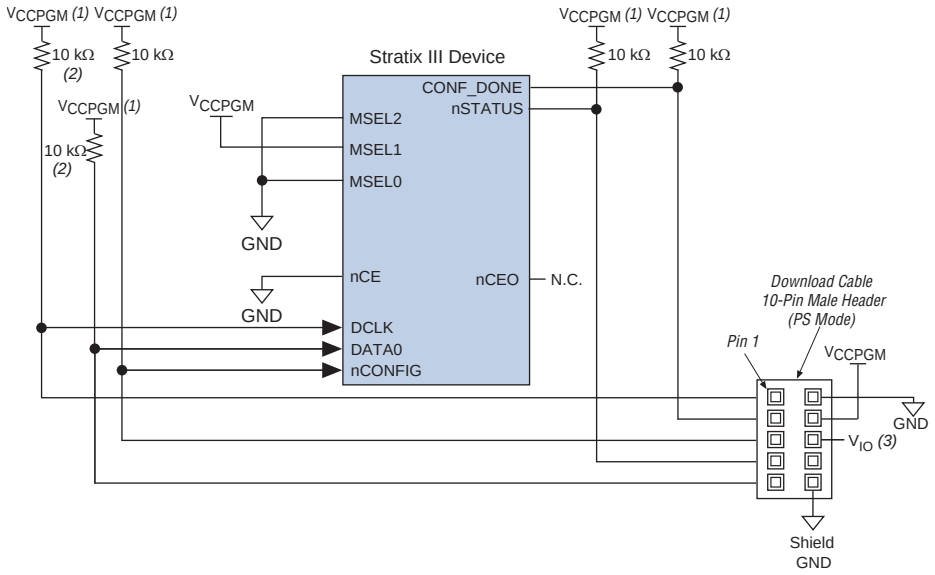
To begin configuration, power the V_{CC}, V_{CCIO}, V_{CCPGM}, and V_{CCPD} voltages (for the banks where the configuration and JTAG pins reside) to the appropriate voltage levels.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. After nSTATUS is released, the device is ready to receive configuration data and the configuration stage begins. The programming hardware or download cable then places the configuration data one bit at a time on the device's DATA0 pin. The configuration data is clocked into the target device until CONF_DONE goes high. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

When using a download cable, setting the **Auto-restart configuration after error** option does not affect the configuration cycle because you must manually restart configuration in the Quartus II software when an error occurs. Additionally, the **Enable user-supplied start-up clock** (**CLKUSR**) option has no affect on the device initialization since this option is disabled in the SOF when programming the device using the Quartus II programmer and download cable. Therefore, if you turn on the CLKUSR option, you do not need to provide a clock on CLKUSR when you are configuring the device with the Quartus II programmer and a download cable.

Figure 11–17 shows PS configuration for Stratix III devices using a USB-Blaster, MasterBlaster, ByteBlaster II, ByteBlasterMV, or EthernetBlaster cable.

Figure 11–17. PS Configuration Using a Download Cable



Notes to Figure 11–17:

(1) You should connect the pull-up resistor to the same supply voltage (V_{CCPGM}) as the USB-Blaster, MasterBlaster (V_{IO} pin), ByteBlaster II, ByteBlasterMV, or EthernetBlaster cable.

(2) You only need the pull-up resistors on DATA0 and DCLK if the download cable is the only configuration scheme used on your board. This ensures that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, you do not need the pull-up resistors on DATA0 and DCLK.

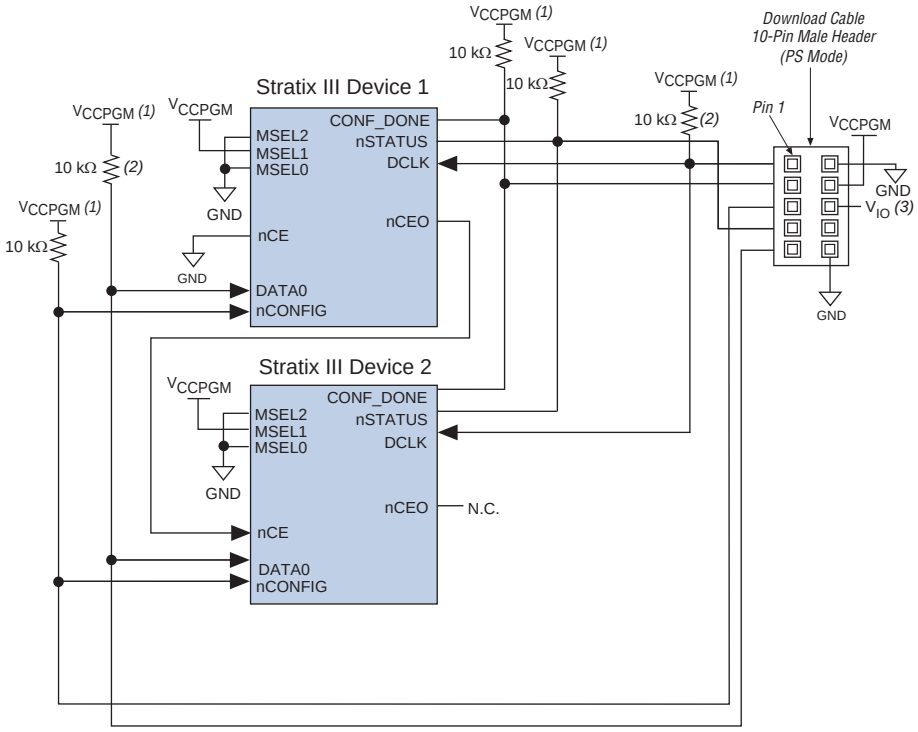
(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCPGM}. Refer to the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value. In the USB-Blaster, ByteBlaster II, ByteBlasterMV, or EthernetBlaster cable, this pin is a no connect.

You can use a download cable to configure multiple Stratix III devices by connecting each device's nCEO pin to the subsequent device's nCE pin. The first device's nCE pin is connected to GND while its nCEO pin is connected to the nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. Because all CONF_DONE pins are tied together, all devices in the chain initialize and enter user mode at the same time.

In addition, because the nSTATUS pins are tied together, the entire chain halts configuration if any device detects an error. The **Auto-restart configuration after error** option does not affect the configuration cycle because you must manually restart configuration in the Quartus II software when an error occurs.

Figure 11–18 shows how to configure multiple Stratix III devices with a download cable.

Figure 11–18. Multi-Device PS Configuration using a Download Cable



Notes to Figure 11–18:

(1) Connect the pull-up resistor to the same supply voltage (V_{CCPGM}) as the USB-Blaster, MasterBlaster (V_{IO} pin), ByteBlaster II, ByteBlasterMV, or EthernetBlaster cable.

(2) You only need the pull-up resistors on DATA0 and DCLK if the download cable is the only configuration scheme used on your board. This is to ensure that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, you do not need the pull-up resistors on DATA0 and DCLK.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCPGM}. Refer to the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value. In the USB-Blaster, ByteBlaster II, ByteBlasterMV, or EthernetBlaster cable, this pin is a no connect.

For more information about how to use the USB-Blaster, MasterBlaster, ByteBlaster II, ByteBlasterMV, or EthernetBlaster cable, refer to the following user guides:

- *USB-Blaster USB Port Download Cable User Guide*
- *MasterBlaster Serial/USB Communications Cable User Guide*
- *ByteBlaster II Parallel Port Download Cable User Guide*
- *ByteBlasterMV Parallel Port Download Cable User Guide*
- *EthernetBlaster Download Cable User Guide*

JTAG Configuration

The JTAG has developed a specification for boundary-scan testing. This boundary-scan test (BST) architecture offers the capability to efficiently test components on PCBs with tight lead spacing. The BST architecture can test pin connections without using physical test probes and capture functional data while a device is operating normally. You can also use the JTAG circuitry to shift configuration data into the device. The Quartus II software automatically generates SOFs that can be used for JTAG configuration with a download cable in the Quartus II software programmer.

For more information about JTAG boundary-scan testing and commands available using Stratix III devices, refer to the following documents:

- *IEEE 1149.1 (JTAG) Boundary Scan Testing in Stratix III Device* chapter of the *Stratix III Device Handbook*
- *AN 425: Using the Command-Line Jam STAPL Solution for Device Programming*

Stratix III devices are designed such that JTAG instructions have precedence over any device configuration modes. Therefore, JTAG configuration can take place without waiting for other configuration modes to complete. For example, if you attempt JTAG configuration of Stratix III devices during PS configuration, PS configuration is terminated and JTAG configuration begins.

You cannot use the Stratix III decompression or design security features if you are configuring your Stratix III device when using JTAG-based configuration.

A device operating in JTAG mode uses four required pins, `TDI`, `TDO`, `TMS`, and `TCK`, and one optional pin, `TRST`. The `TCK` pin has an internal weak pull-down resistor, while the `TDI`, `TMS`, and `TRST` pins have weak internal pull-up resistors (typically 25 kΩ). JTAG output pin `TDO` and all JTAG input pins are powered by the 2.5 V/3.0 V/3.3 V V_{CCPD} power supply of I/O bank 1A. All the JTAG pins support only LVTTL I/O standard.

All user I/O pins are tri-stated during JTAG configuration. Table 11–11 explains each JTAG pin's function.

The `TDO` output is powered by the V_{CCPD} power supply of I/O bank 1A. For recommendations on how to connect a JTAG chain with multiple voltages across the devices in the chain, refer to the *IEEE 1149.1 (JTAG) Boundary Scan Testing in Stratix III Devices* chapter of the *Stratix III Device Handbook*.

Table 11–11. Dedicated JTAG Pins

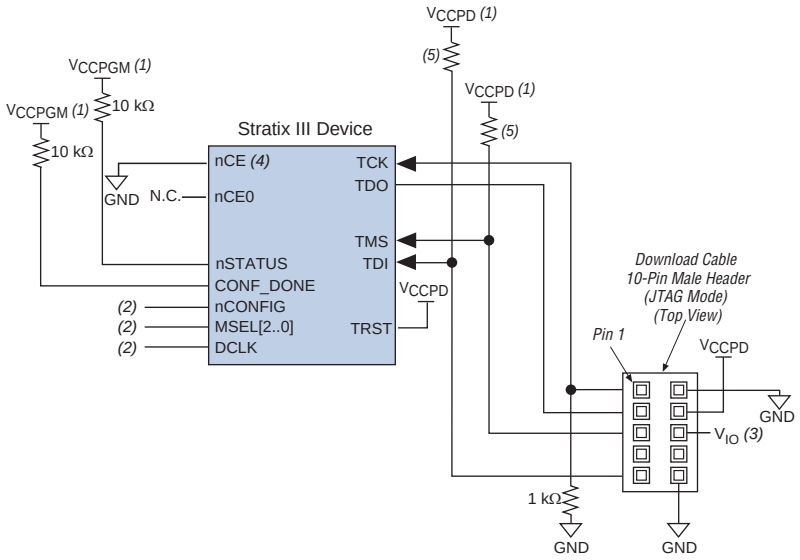
Pin Name	Pin Type	Description
TDI	Test data input	Serial input pin for instructions as well as test and programming data. Data is shifted in the rising edge of TCK. If the JTAG interface is not required on the board, you can disable the JTAG circuitry by connecting this pin to V_{CCPD}.
TDO	Test data output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. If the JTAG interface is not required on the board, you can disable the JTAG circuitry by leaving this pin unconnected.
TMS	Test mode select	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, you must set up TMS before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. If the JTAG interface is not required on the board, you can disable the JTAG circuitry by connecting this pin to VCCPD.
TCK	Test clock input	The clock input to the BST circuitry. Some operations occur at the rising edge while others occur at the falling edge. If the JTAG interface is not required on the board, you can disable the JTAG circuitry by connecting this pin to GND.
TRST	Test reset input (optional)	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to IEEE Std. 1149.1. If the JTAG interface is not required on the board, you can disable the JTAG circuitry by connecting this pin to GND.

During JTAG configuration, you can download data to the device on the PCB through the USB-Blaster, MasterBlaster, ByteBlaster II, ByteBlasterMV, or EthernetBlaster download cables. Configuring devices through a cable is similar to programming devices in-system, except you should connect the TRST pin to V_{CCPD}. This ensures that the TAP controller is not reset.

The JRunner™ software driver is developed to configure Altera FPGA devices in JTAG mode through the ByteBlaster II or ByteBlasterMV download cables for embedded configurations. For more information, refer to *AN 414: The JRunner Software Driver*.

Figure 11–19 shows JTAG configuration of a single Stratix III device.

Figure 11–19. JTAG Configuration of a Single Device Using a Download Cable



Notes to Figure 11–19:

(1) You should connect the pull-up resistor to the same supply voltage as the USB-Blaster, MasterBlaster (VIO pin), ByteBlaster II, ByteBlasterMV, or EthernetBlaster cables. The voltage supply can be connected to the V_{CCPD} of the device.

(2) You should connect the nCONFIG and MSEL[2..0] pins to support a non-JTAG configuration scheme. If you only use the JTAG configuration, connect nCONFIG to V_{CCPGM}, and MSEL[2..0] to ground. Pull DCLK either high or low, whichever is convenient on your board.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCPD}. Refer to the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value. In the USB-Blaster, ByteBlaster II, ByteBlasterMV, and EthernetBlaster, this pin is a no connect.

(4) You must connect nCE to GND or drive it low for successful JTAG configuration.

(5) Pull-up resistor values can vary from 1 kΩ to 10 kΩ

To configure a single device in a JTAG chain, the programming software places all other devices in bypass mode. In bypass mode, devices pass programming data from the TDI pin to the TDO pin through a single bypass register without being affected internally. This scheme enables the programming software to program or verify the target device. Configuration data driven into the device appears on the TDO pin one clock cycle later.

The Quartus II software verifies successful JTAG configuration upon completion. At the end of configuration, the software checks the state of CONF_DONE through the JTAG port. When Quartus II generates a JAM file (.**jam)** for a multi-device chain, it contains instructions so that all the devices in the chain will be initialized at the same time. If CONF_DONE is not high, the Quartus II software indicates that configuration has failed. If CONF_DONE is high, the software indicates that configuration was successful. After the configuration bitstream is transmitted serially through the JTAG TDI port, the TCK port is clocked an additional 1,094 cycles to perform device initialization.

Stratix III devices have dedicated JTAG pins that always function as JTAG pins. Not only can you perform JTAG testing on Stratix III devices before and after, but also during configuration. While other device families do not support JTAG testing during configuration, Stratix III devices support the bypass, id code, and sample instructions during configuration without interrupting configuration. All other JTAG instructions may only be issued by first interrupting configuration and reprogramming I/O pins using the CONFIG_IO instruction.

The CONFIG_IO instruction allows I/O buffers to be configured by using the JTAG port and when issued, interrupts configuration. This instruction allows you to perform board-level testing prior to configuring the Stratix III device or waiting for a configuration device to complete configuration. When configuration has been interrupted and JTAG testing is complete, you must reconfigure the part by using JTAG (PULSE_CONFIG instruction) or by pulsing nCONFIG low.

The chip-wide reset (DEV_CLRn) and chip-wide output enable (DEV_OE) pins on Stratix III devices do not affect JTAG boundary-scan or programming operations. Toggling these pins does not affect JTAG operations (other than the usual boundary-scan operation).

When designing a board for JTAG configuration of Stratix III devices, consider the dedicated configuration pins. Table 11–12 lists how these pins should be connected during JTAG configuration.

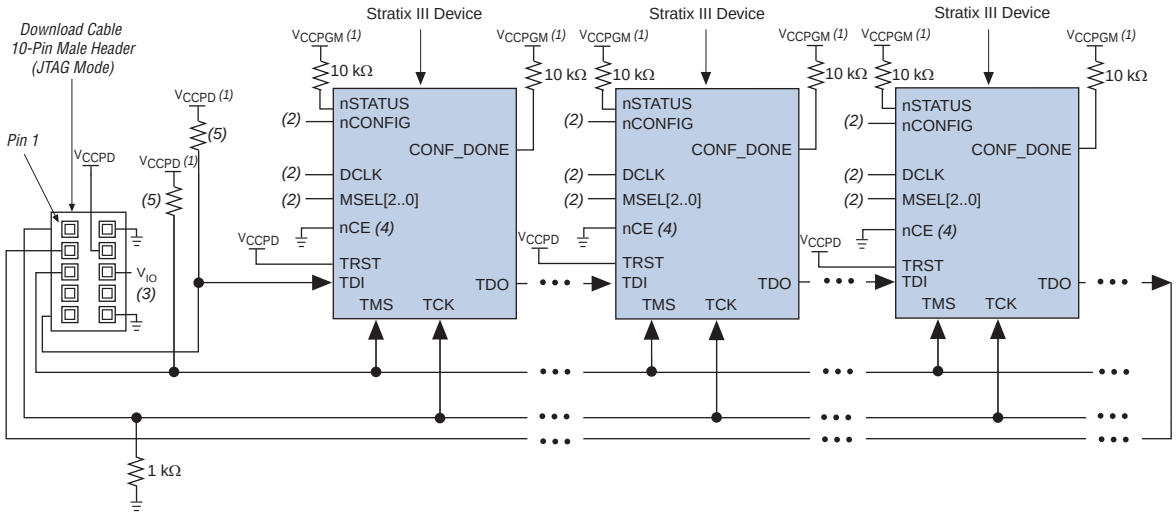
Table 11–12. Dedicated Configuration Pin Connections During JTAG Configuration

Signal	Description
nCE	On all Stratix III devices in the chain, nCE should be driven low by connecting it to ground, pulling it low by using a resistor, or driving it by some control circuitry. For devices that are also in multi-device FPP, AS, or PS configuration chains, the nCE pins should be connected to GND during JTAG configuration or JTAG configured in the same order as the configuration chain.
nCEO	On all Stratix III devices in the chain, you can leave nCEO floating or connected to the nCE of the next device.
MSEL	These pins must not be left floating. These pins support whichever non-JTAG configuration is used in production. If you only use JTAG configuration, tie these pins to ground.
nCONFIG	Driven high by connecting to V_{CCPGM}, pull up by using a resistor, or driven high by some control circuitry.
nSTATUS	Pull to V_{CCPGM} by using a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each nSTATUS pin should be pulled up to V_{CCPGM} individually.
CONF_DONE	Pull to V_{CCPGM} by using a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each CONF_DONE pin should be pulled up to V_{CCPGM} individually. CONF_DONE going high at the end of JTAG configuration indicates successful configuration.
DCLK	Should not be left floating. Drive low or high, whichever is more convenient on your board.

When programming a JTAG device chain, one JTAG-compatible header is connected to several devices. The number of devices in the JTAG chain is limited only by the drive capability of the download cable. When four or more devices are connected in a JTAG chain, Altera recommends buffering the TCK, TDI, and TMS pins with an on-board buffer.

JTAG-chain device programming is ideal when the system contains multiple devices, or when testing your system using JTAG BST circuitry. Figure 11–20 shows multi-device JTAG configuration.

Figure 11–20. JTAG Configuration of Multiple Devices Using a Download Cable



Notes to Figure 11–20:

(1) Connect the pull-up resistor to the same supply voltage as the USB-Blaster, MasterBlaster (V_{IO} pin), ByteBlaster II, ByteBlasterMV, or EthernetBlaster cables. The voltage supply can be connected to the V_{CCPD} of the device.

(2) Connect the nCONFIG, MSEL[2..0] pins to support a non-JTAG configuration scheme. If you only use JTAG configuration, connect nCONFIG to V_{CCPGM}, and MSEL[2..0] to ground. Pull DCLK either high or low, whichever is convenient on your board.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCPD}. Refer to the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value. In the USB-Blaster, ByteBlaster II, ByteBlasterMV, and EthernetBlaster, this pin is a no connect.

(4) You must connect nCE to GND or drive it low for successful JTAG configuration.

(5) Pull-up resistor values can vary from 1 kΩ to 10 kΩ

You must connect the nCE pin to GND or drive it low during JTAG configuration. In multi-device FPP, AS, and PS configuration chains, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. In addition, the CONF_DONE and nSTATUS signals are all shared in multi-device FPP, AS, or PS configuration chains so the devices can enter user mode at the same time after configuration is complete. When the CONF_DONE and nSTATUS signals are shared among all the devices, you must configure every device when JTAG configuration is performed.

If you only use JTAG configuration, Altera recommends that you connect the circuitry as shown in Figure 11–20, where each of the CONF_DONE and nSTATUS signals are isolated, so that each device can enter user mode individually.

After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. Therefore, if these devices are also in a JTAG chain, make sure the nCE pins are connected to GND during JTAG configuration or that the devices are JTAG configured in the same order as the configuration chain. As long as the devices are JTAG configured in the same order as the multi-device configuration chain, the nCEO of the previous device will drive the nCE of the next device low when it has successfully been JTAG configured.

You can place other Altera devices that have JTAG support in the same JTAG chain for device programming and configuration.

JTAG configuration support has been enhanced and allows more than 17 Stratix III devices to be cascaded in a JTAG chain.

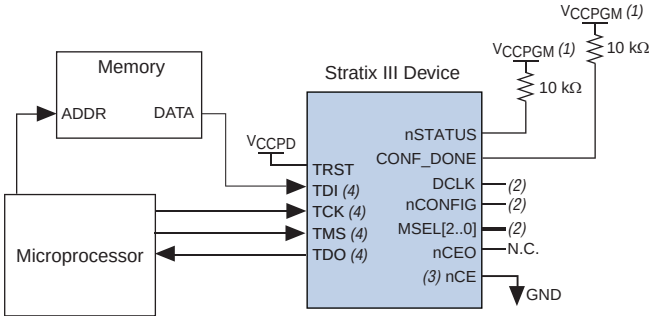
For more information about configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera Device Chains* chapter in the *Configuration Handbook*.

You can configure Stratix III devices using multiple configuration schemes on the same board. Combining JTAG configuration with passive serial (PS) or active serial (AS) configuration on your board is useful in the prototyping environment because it allows multiple methods to configure your FPGA.

For more information about combining JTAG configuration with other configuration schemes, refer to the *Combining Different Configuration Schemes* chapter in the Configuration Handbook.

Figure 11–21 shows JTAG configuration of a Stratix III device with a microprocessor.

Figure 11–21. JTAG Configuration of a Single Device Using a Microprocessor



Notes to Figure 11–21:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for all devices in the chain. V_{CCPGM} should be high enough to meet the V_{IH} specification of the I/O on the device.

(2) Connect the `nCONFIG` and `MSEL[2..0]` pins to support a non-JTAG configuration scheme. If you use only the JTAG configuration, connect `nCONFIG` to V_{CCPGM}, and `MSEL[2..0]` to ground. Pull `DCLK` either high or low, whichever is convenient on your board.

(3) You must connect `nCE` to `GND` or drive it low for successful JTAG configuration.

(4) Microprocessor should use the same I/O standard as V_{CCPD} to drive the JTAG pins.

Jam STAPL

Jam STAPL, JEDEC standard JESD-71, is a standard file format for in-system programmability (ISP) purposes. Jam STAPL supports programming or configuration of programmable devices and testing of electronic systems, using the IEEE 1149.1 JTAG interface. Jam STAPL is a freely licensed open standard.

The Jam Player provides an interface for manipulating the IEEE Std. 1149.1 JTAG TAP state machine.

For more information about JTAG and Jam STAPL in embedded environments, refer to *AN 425: Using Command-Line Jam STAPL Solution for Device Programming*. To download the jam player, visit the Altera web site at www.altera.com.

Device Configuration Pins

The following tables describe the connections and functionality of all the configuration-related pins on the Stratix III devices. Table 11–13 summarizes the Stratix III configuration pins and their power supply.

Table 11–13. Stratix III Configuration Pin Summary *(Note 1)* (Part 1 of 2)

Description	Input/Output	Dedicated	Powered By	Configuration Mode
TDI	Input	Yes	V_{CCPD} *(2)*	JTAG
TMS	Input	Yes	V_{CCPD} *(2)*	JTAG
TCK	Input	Yes	V_{CCPD} *(2)*	JTAG
TRST	Input	Yes	V_{CCPD} *(2)*	JTAG
TDO	Output	Yes	V_{CCPD} *(2)*	JTAG
CRC_ERROR	Output	—	Pull-up	Optional, all modes
DATA0	Input	—	V_{CCPGM}/V_{CCIO} *(3)*	All modes except JTAG
DATA[7..1]	Input	—	V_{CCPGM}/V_{CCIO} *(3)*	FPP
INIT_DONE	Output	—	Pull-up	Optional, all modes
CLKUSR	Input	—	V_{CCPGM}/V_{CCIO} *(3)*	Optional
nSTATUS	bi-directional	Yes	Pull-up	All modes
nCE	Input	Yes	V_{CCPGM}	All modes
CONF_DONE	bi-directional	Yes	Pull-up	All modes
nCONFIG	Input	Yes	V_{CCPGM}	All modes
PORSEL	Input	Yes	V_{CCPGM}	All modes
ASDO	Output	Yes	V_{CCPGM}	AS
nCSO	Output	Yes	V_{CCPGM}	AS
DCLK	Input	Yes	V_{CCPGM}	PS, FPP
—	Output	—	V_{CCPGM}	AS
nIO_PULLUP	Input	Yes	V_{CCPGM}	All modes
nCEO	Output	Yes	V_{CCPGM}	All modes

Table 11–13. Stratix III Configuration Pin Summary *(Note 1)* (Part 2 of 2)

Description	Input/Output	Dedicated	Powered By	Configuration Mode
MSEL[2..0]	Input	Yes	V_{CCPGM}	All modes

Notes to Table 11–13:

(1) The total number of pins is 30. The total number of dedicated pins is 19.

(2) The JTAG output pin TDO and all JTAG input pins are powered by the 2.5 V/3.0 V/3.3-V V_{CCPD} power supply of I/O bank 1A.

(3) These dual purpose pins are powered by V_{CCPGM} during configuration, then are powered by V_{CCIO} while in user mode.This applies for all configuration modes.

Table 11–14 describes the dedicated configuration pins, which are required to be connected properly on your board for successful configuration. Some of these pins may not be required for your configuration schemes.

Table 11–14. Dedicated Configuration Pins on the Stratix III Device (Part 1 of 5)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
VCCPGM	N/A	All	Power	Dedicated power pin. Use this pin to power all dedicated configuration inputs, dedicated configuration outputs, dedicated configuration bi-direction pins, and some of the dual functional pins that are used for configuration. You must connect this pin to 1.8-V, 2.5-V, 3.0-V, or 3.3-V. V_{CCPGM} must ramp-up from 0-V to 3.3-V within 100 ms. If V_{CCPGM} is not ramped up within this specified time, your Stratix III device will not configure successfully. If your system does not allow for a VCCPGM ramp-up time of 100 ms or less, you must hold nCONFIG low until all power supplies are stable.
VCCPD	N/A	All	Power	Dedicated power pin. Use this pin to power the I/O pre-drivers, the JTAG input and output pins, and the design security circuitry. You must connect this pin to 2.5-V, 3.0-V, or 3.3-V depending on the I/O standards selected. For 3.3-V I/O standards, VCCPD=3.3-V, for 3.0-V I/O standards, V_{CCPD} = 3.0 V, for 2.5-V or below I/O standards, V_{CCPD} = 2.5 V. V_{CCPD} must ramp-up from 0-V to 2.5-V / 3.0-V/3.3-V within 100 ms. If V_{CCPD} is not ramped up within this specified time, your Stratix III device will not configure successfully. If your system does not allow for a V_{CCPD} ramp-up time of 100 ms or less, you must hold nCONFIG low until all power supplies are stable.
PORSEL	N/A	All	Input	Dedicated input which selects either a POR time of 12 ms or 100 ms. A logic high (1.8 V, 2.5 V, 3.0 V, 3.3 V) selects a POR time of approximately 12 ms and a logic low selects a POR time of approximately 100 ms. The PORSEL input buffer is powered by VCCPGM and has an internal 5-kΩ pull-down resistor that is always active. You should tie the PORSEL pin directly to VCCPGM or GND.

Table 11–14. Dedicated Configuration Pins on the Stratix III Device (Part 2 of 5)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nIO_PULLUP	N/A	All	Input	Dedicated input that chooses whether the internal pull-up resistor on the user I/O pins and dual-purpose I/O pins (nCSO, nASDO, DATA[7..0], nWS, nRS, RDYnBSY, nCS, CLKUSR, INIT_DONE) are on or off before and during configuration. A logic high (1.8 V, 2.5 V, 3.0 V, 3.3 V) turns off the weak internal pull-up resistors, while a logic low turns them on. The nIO-PULLUP input buffer is powered by VCCPGM and has an internal 5-kΩ pull-down resistor that is always active. You can tie the nIO-PULLUP directly to VCCPGM or use a 1-kΩ pull-up resistor or tie it directly to GND.
MSEL[2..0]	N/A	All	Input	3-bit configuration input that sets the Stratix III device configuration scheme. Refer to Table 11–1 for the appropriate connections. You must hard-wire these pins to VCCPGM or GND. The MSEL[2..0] pins have internal 5-kΩ pull-down resistors that are always active.
nCONFIG	N/A	All	Input	Configuration control input. Pulling this pin low during user-mode will cause the device to lose its configuration data, enter a reset state, and tri-state all I/O pins. Returning this pin to a logic high level will initiate a reconfiguration. Configuration is possible only if this pin is high, except in JTAG programming mode when nCONFIG is ignored.

Table 11–14. Dedicated Configuration Pins on the Stratix III Device (Part 3 of 5)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nSTATUS	N/A	All	Bi-directional open-drain	The device drives nSTATUS low immediately after power-up and releases it after the POR time. During user mode and regular configuration, this pin is pulled high by an external 10-kΩ resistor. This pin, when driven low by Stratix III, indicates that the device is being initialized and has encountered an error during configuration. Status output. If an error occurs during configuration, nSTATUS is pulled low by the target device. Status input. If an external source drives the nSTATUS pin low during configuration or initialization, the target device enters an error state. Driving nSTATUS low after configuration and initialization does not affect the configured device. If you use a configuration device, driving nSTATUS low will cause the configuration device to attempt to configure the device, but since the device ignores transitions on nSTATUS in user-mode, the device does not reconfigure. To initiate a reconfiguration, nCONFIG must be pulled low. If you have enabled the **Auto-restart configuration after error** option, the nSTATUS pin transitions from high to low and back again to high when a configuration error is detected. This appears as a low pulse at the nSTATUS pin with a minimum pulse width of 10 µs to a maximum pulse width of 500 µs, as defined in the t_{STATUS} specification.
nSTATUS (continued)	—	—	—	If V_{CCPGM} and V_{CCIO} are not fully powered up, the following could occur: ■ V_{CCPGM} and V_{CCIO} are powered high enough for the nSTATUS buffer to function properly, and nSTATUS is driven low. When V_{CCPGM} and V_{CCIO} are ramped up, POR trips and nSTATUS is released after POR expires. ■ V_{CCPGM} and V_{CCIO} are not powered high enough for the nSTATUS buffer to function properly. In this situation, nSTATUS might appear logic high, triggering a configuration attempt that would fail because POR did not yet trip. When V_{CCPD} and V_{CCIO} are powered up, nSTATUS is pulled low because POR did not yet trip. When POR trips after V_{CCPGM} and V_{CCIO} are powered up, nSTATUS is released and pulled high. At that point, reconfiguration is triggered and the device is configured.

Table 11–14. Dedicated Configuration Pins on the Stratix III Device (Part 4 of 5)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
CONF_DONE	N/A	All	Bi-directional open-drain	Status output. The target device drives the CONF_DONE pin low before and during configuration. After all configuration data is received without error and the initialization cycle starts, the target device releases CONF_DONE. Status input. After all data is received and CONF_DONE goes high, the target device initializes and enters user mode. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize. Driving CONF_DONE low after configuration and initialization does not affect the configured device. Do not connect bus holds or ADC to CONF_DONE pin.
nCE	N/A	All	Input	Active-low chip enable. The nCE pin activates the device with a low signal to allow configuration. The nCE pin must be held low during configuration, initialization, and user mode. In single device configuration, it should be tied low. In multi-device configuration, nCE of the first device is tied low while its nCEO pin is connected to nCE of the next device in the chain. The nCE pin must also be held low for successful JTAG programming of the device.
nCEO	N/A	All	Output	Output that drives low when device configuration is complete. In single device configuration, this pin is left floating. In multi-device configuration, this pin feeds the next device's nCE pin. The nCEO of the last device in the chain is left floating. The nCEO pin is powered by V_{CCPGM}.
ASDO *(1)*	N/A	AS	Output	Control signal from the Stratix III device to the serial configuration device in AS mode used to read out configuration data. In AS mode, ASDO has an internal pull-up resistor that is always active.
nCSO *(1)*	N/A	AS	Output	Output control signal from the Stratix III device to the serial configuration device in AS mode that enables the configuration device. In AS mode, nCSO has an internal pull-up resistor that is always active.

Table 11–14. Dedicated Configuration Pins on the Stratix III Device (Part 5 of 5)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
DCLK *(1)*	N/A	Synchronous configuration schemes (PS, FPP, AS)	Input (PS, FPP) Output (AS)	DCLK has an internal pull-up resistor (typically 25 kΩ) that is always active. In AS mode, DCLK is an output from the Stratix III device that provides timing for the configuration interface. After AS configuration, this pin is driven to an inactive state. In schemes that use a configuration device, DCLK will be driven low after configuration is done. In schemes that use a control host, DCLK should be driven either high or low, whichever is more convenient. Toggling this pin after configuration does not affect the configured device.
DATA0 *(1)*	N/A in AS mode. I/O in PS or FPP mode	PS, FPP, AS	Input	Data input. In serial configuration modes, bit-wide configuration data is presented to the target device on the DATA0 pin. In AS mode, DATA0 has an internal pull-up resistor that is always active. After PS or FPP configuration, DATA0 is available as a user I/O pin and the state of this pin depends on the **Dual-Purpose Pin** settings.
DATA[7..1]	I/O	Parallel configuration schemes (FPP)	Inputs	Data inputs. Byte-wide configuration data is presented to the target device on DATA[7..0]. In serial configuration schemes, they function as user I/O pins during configuration, which means they are tri-stated. After configuration, DATA[7..1] are available as user I/O pins and the state of these pin depends on the **Dual-Purpose Pin** settings.

Note to Table 11–14:

(1) To tri-state AS configuration pins in AS configuration scheme, turn on **Enable input tri-state on active configuration pins in user mode** option from the **Device and Pin Options** dialog box. This tri-states DCLK, nCSO, Data0, and ASDO pins. Dual-purpose Pins Setting for Data0 is ignored. To set Data0 to a different setting, for example to use Data0 pin as a regular I/O in user mode, turn off **Enable input tri-state on active configuration pins in user mode** option and set your desired setting from the Dual-purpose Pins Setting menu.

Table 11–15 describes the optional configuration pins. If these optional configuration pins are not enabled in the Quartus II software, they are available as general-purpose user I/O pins. Therefore, during configuration, these pins function as user I/O pins and are tri-stated with weak pull-up resistors.

Table 11–15. Optional Configuration Pins

Pin Name	User Mode	Pin Type	Description
CLKUSR	N/A if option is on. I/O if option is off.	Input	Optional user-supplied clock input synchronizes the initialization of one or more devices. Enable this pin by turning on the **Enable user-supplied start-up clock** (**CLKUSR**) option in the Quartus II software.
INIT_DONE	N/A if option is on. I/O if option is off.	Output open-drain	Use the Status pin to indicate when the device has initialized and is in user mode. When nCONFIG is low and during the beginning of configuration, the INIT_DONE pin is tri-stated and pulled high due to an external 10-kΩ pull-up resistor. After the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low. When initialization is complete, the INIT_DONE pin is released and pulled high and the device enters user mode. Thus, the monitoring circuitry must be able to detect a low-to-high transition. This pin is enabled by turning on the **Enable INIT_DONE output** option in the Quartus II software.
DEV_OE	N/A if option is on. I/O if option is off.	Input	Optional pin that allows you to override all tri-states on the device. When this pin is driven low, all I/O pins are tri-stated, when this pin is driven high, all I/O pins behave as programmed. Enable this pin by turning on the **Enable device-wide output enable** (**DEV_OE**) option in the Quartus II software.
DEV_CLRn	N/A if option is on. I/O if option is off.	Input	Optional pin that allows you to override all clears on all device registers. When this pin is driven low, all registers are cleared. When this pin is driven high, all registers behave as programmed. This pin is enabled by turning on the **Enable device-wide reset** (**DEV_CLRn**) option in the Quartus II software.

Table 11–16 describes the dedicated JTAG pins. JTAG pins must be kept stable before and during configuration to prevent accidental loading of JTAG instructions. The TDI, TMS, and TRST have weak internal pull-up resistors while TCK has a weak internal pull-down resistor (typically 25 kΩ). If you plan to use the SignalTap® embedded logic array analyzer, you must connect the JTAG pins of the Stratix III device to a JTAG header on your board.

Table 11–16. Dedicated JTAG Pins

Pin Name	User Mode	Pin Type	Description
TDI	N/A	Input	Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. The TDI pin is powered by the 2.5-V/3.0-V/3.3-V V_{CCPD} supply. If the JTAG interface is not required on the board, you can disable the JTAG circuitry by connecting this pin to V_{CCPD}.
TDO	N/A	Output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. The TDO pin is powered by V_{CCPD} For recommendations on connecting a JTAG chain with multiple voltages across the devices in the chain, refer to the chapter *IEEE 1149.1 (JTAG) Boundary Scan Testing in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*. If the JTAG interface is not required on the board, you can disable the JTAG circuitry by leaving this pin unconnected.
TMS	N/A	Input	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. The TMS pin is powered by the 2.5-V/3.0-V/3.3-V V_{CCPD}. If the JTAG interface is not required on the board, you can disable the JTAG circuitry by connecting this pin to V_{CCPD}.
TCK	N/A	Input	The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. The TCK pin is powered by the 2.5-V/3.0-V/3.3-V V_{CCPD} supply. It is expected that the clock input waveform have a nominal 50% duty cycle. If the JTAG interface is not required on the board, you can disable the JTAG circuitry by connecting TCK to GND.
TRST	N/A	Input	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to IEEE Std. 1149.1. The TRST pin is powered by the 2.5-V/3.0-V/3.3-V V_{CCPD} supply. You should hold TMS at 1 or you should keep TCK static while TRST is changed from 0 to 1. If the JTAG interface is not required on the board, you can disable the JTAG circuitry by connecting the TRST pin to GND.

Chapter Revision History

Table 11–17 lists the revision history for this chapter.

Table 11–17. Chapter Revision History (Part 1 of 2)

Date	Version	Changes Made
March 2011	2.0	■ Updated the "FPP Configuration Using a MAX II Device as an External Host", "Fast Active Serial Configuration (Serial Configuration Devices)", and "PS Configuration Using a MAX II Device as an External Host" ■ Updated Table 11–14.
July 2010	1.9	■ Updated Table 11–14. ■ Updated "FPP Configuration Using a MAX II Device as an External Host" on page 11–8.
March 2010	1.8	Updated for the Quartus II software version 9.1 SP2 release: ■ Added Figure 11–11. ■ Updated Figure 11–6, Figure 11–7, Figure 11–16, Figure 11–19, and Figure 11–20. ■ Updated "Estimating Active Serial Configuration Time" section. ■ Added Table 11–8. ■ Updated Table 11–8, Table 11–13, and Table 11–14. ■ Removed "Conclusion" section.
May 2009	1.7	■ Updated Table 11–1, Table 11–2, Table 11–5, Table 11–6, Table 11–9, and Table 11–13. ■ Updated Figure 11–6, Figure 11–16, Figure 11–17, Figure 11–18, Figure 11–19, and Figure 11–20. ■ Updated "PS Configuration Using a Microprocessor", "PS Configuration Using a Download Cable", and "JTAG Configuration" sections. ■ Removed Figure 11-12 Fast AS Configuration Timing. ■ Removed Table 11-8 Fast AS Timing Parameters for Stratix III devices.
February 2009	1.6	■ Updated Figure 11–6, Figure 11–7, Figure 11–12, and Figure 11–16. ■ Removed "Referenced Documents" section.
October 2008	1.5	■ Updated "FPP Configuration Using a MAX II Device as an External Host", "Fast Active Serial Configuration (Serial Configuration Devices)", "JTAG Configuration", "Power-On Reset Circuit", "PS Configuration Using a MAX II Device as an External Host", and "PS Configuration Using a Download Cable" sections. ■ Updated Table 11–13 and Table 11–14. ■ Updated New Document Format. ■ Updated (Note 3) to Figure 11–17. ■ Updated (Note 3) to Figure 11–18. ■ Updated (Note 3) to Figure 11–19. ■ Updated (Note 3) to Figure 11–20.

Table 11–17. Chapter Revision History (Part 2 of 2)

Date	Version	Changes Made
May 2008	1.4	■ Updated "Power-On Reset Circuit" section. ■ Updated "Fast Active Serial Configuration (Serial Configuration Devices)" section. ■ Updated Table 11–4, Table 11–11, Table 11–12, Table 11–13, Table 11–14, and Table 11–16. ■ Updated Figure 11–3, Figure 11–4, Figure 11–5, Figure 11–6, Figure 11–7, Figure 11–8, Figure 11–9, Figure 11–10, Figure 11–11, Figure 11–13, Figure 11–14, Figure 11–15, Figure 11–17, Figure 11–18, Figure 11–19, Figure 11–20, and Figure 11–21.
November 2007	1.3	■ Updated Table 11–2. ■ Updated Figure 11–8, Figure 11–14, Figure 11–15, Figure 11–17, and Figure 11–18.
October 2007	1.2	■ Updated Table 11–13, Table 11–14. ■ Updated Figure 11–6, Figure 11–7, Figure 11–9, Figure 11–10, Figure 11–11, and Figure 11–13. ■ Removed text regarding enhanced configuration device support. Removed Figure 11–19. ■ Added live links for references. ■ Added section "Referenced Documents"
May 2007	1.1	Removed Bank Column from Table 11–13.
November 2006	1.0	Initial Release

7. Configuring Stratix II and Stratix II GX Devices

SII52007-4.5

Introduction

Stratix® II and Stratix II GX devices use SRAM cells to store configuration data. Because SRAM memory is volatile, configuration data must be downloaded to Stratix II and Stratix II GX devices each time the device powers up. Stratix II and Stratix II GX devices can be configured using one of five configuration schemes: the fast passive parallel (FPP), active serial (AS), passive serial (PS), passive parallel asynchronous (PPA), and Joint Test Action Group (JTAG) configuration schemes. All configuration schemes use either an external controller (for example, a MAX® II device or microprocessor) or a configuration device.

Configuration Devices

The Altera enhanced configuration devices (EPC16, EPC8, and EPC4) support a single-device configuration solution for high-density devices and can be used in the FPP and PS configuration schemes. They are ISP-capable through its JTAG interface. The enhanced configuration devices are divided into two major blocks, the controller and the flash memory.

For information on enhanced configuration devices, refer to the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* in volume 2 of the *Configuration Handbook*.

The Altera serial configuration devices (EPCS64, EPCS16, and EPCS4) support a single-device configuration solution for Stratix II and Stratix II GX devices and are used in the AS configuration scheme. Serial configuration devices offer a low cost, low pin count configuration solution.

For information on serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

The EPC2 configuration devices provide configuration support for the PS configuration scheme. The EPC2 device is ISP-capable through its JTAG interface. The EPC2 device can be cascaded to hold large configuration files.

For more information on EPC2 configuration devices, refer to the *Configuration Devices for SRAM-Based LUT Devices Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

The configuration scheme is selected by driving the Stratix II or Stratix II GX device MSEL pins either high or low as shown in Table 7–1. The MSEL pins are powered by the V_{CCIO} power supply of the bank they reside in. The MSEL[3..0] pins have 9-kΩ internal pull-down resistors that are always active. During power-on reset (POR) and during reconfiguration, the MSEL pins have to be at LVTTL V_{IL} and V_{IH} levels to be considered a logic low and logic high.

To avoid any problems with detecting an incorrect configuration scheme, hard-wire the MSEL[] pins to V_{CCPD} and GND, without any pull-up or pull-down resistors. Do not drive the MSEL[] pins by a microprocessor or another device.

Table 7–1. Stratix II and Stratix II GX Configuration Schemes (Part 1 of 2)

Configuration Scheme	MSEL3	MSEL2	MSEL1	MSEL0
Fast passive parallel (FPP)	0	0	0	0
Passive parallel asynchronous (PPA)	0	0	0	1
Passive serial (PS)	0	0	1	0
Remote system upgrade FPP *(1)*	0	1	0	0
Remote system upgrade PPA *(1)*	0	1	0	1
Remote system upgrade PS *(1)*	0	1	1	0
Fast AS (40 MHz) *(2)*	1	0	0	0
Remote system upgrade fast AS (40 MHz) *(2)*	1	0	0	1
FPP with decompression and/or design security feature enabled *(3)*	1	0	1	1
Remote system upgrade FPP with decompression and/or design security feature enabled *(1)*, *(3)*	1	1	0	0
AS (20 MHz) *(2)*	1	1	0	1
Remote system upgrade AS (20 MHz) *(2)*	1	1	1	0
JTAG-based configuration *(5)*	*(4)*	*(4)*	*(4)*	*(4)*

Table 7–1. Stratix II and Stratix II GX Configuration Schemes (Part 2 of 2)				
Configuration Scheme	**MSEL3**	**MSEL2**	**MSEL1**	**MSEL0**

Notes to Table 7–1:
(1) These schemes require that you drive the RUnLU pin to specify either remote update or local update. For more information about remote system upgrades in Stratix II devices, refer to the *Remote System Upgrades With Stratix II & Stratix II GX Devices* chapter in volume 2 of the *Stratix II Device Handbook* or the *Remote System Upgrades With Stratix II & Stratix II GX Devices* chapter in volume 2 of the *Stratix II GX Device Handbook*.
(2) Only the EPCS16 and EPCS64 devices support up to a 40 MHz DCLK. Other EPCS devices support up to a 20 MHz DCLK. Refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* for more information.
(3) These modes are only supported when using a MAX II device or a microprocessor with flash memory for configuration. In these modes, the host system must output a DCLK that is 4× the data rate.
(4) Do not leave the MSEL pins floating. Connect them to V_{CCPD} or ground. These pins support the non-JTAG configuration scheme used in production. If only JTAG configuration is used, you should connect the MSEL pins to ground.
(5) JTAG-based configuration takes precedence over other configuration schemes, which means MSEL pin settings are ignored.

Stratix II and Stratix II GX devices offer design security, decompression, and remote system upgrade features. Design security using configuration bitstream encryption is available in Stratix II and Stratix II GX devices, which protects your designs. Stratix II and Stratix II GX devices can receive a compressed configuration bit stream and decompress this data in real-time, reducing storage requirements and configuration time. You can make real-time system upgrades from remote locations of your Stratix II and Stratix II GX designs with the remote system upgrade feature.

Table 7–2 and Table 7–3 show the uncompressed configuration file sizes for Stratix II and Stratix II GX devices, respectively.

Table 7–2. Stratix II Uncompressed .rbf Sizes Notes (1), (2)		
Device	**Data Size (Bits)**	**Data Size (MBytes)**
EP2S15	4,721,544	0.590
EP2S30	9,640,672	1.205
EP2S60	16,951,824	2.119
EP2S90	25,699,104	3.212
EP2S130	37,325,760	4.666
EP2S180	49,814,760	6.227

Notes to Table 7–2:
(1) These values are final.
(2) **.rbf**: Raw Binary File.

Table 7–3. Stratix II GX Uncompressed .rbf Sizes *Note (1)*		
Device	**Data Size (Bits)**	**Data Size (MBytes)**
EP2SGX30C EP2SGX30D	9,640,672	1.205
EP2SGX60C EP2SGX60D EP2SGX60E	16,951,824	2.119
EP2SGX90E EP2SGX90F	25,699,104	3.212
EP2SGX130G	37,325,760	4.666

Note to Table 7–3:
(1) **.rbf**: Raw Binary File.

Use the data in Table 7–2 to estimate the file size before design compilation. Different configuration file formats, such as a Hexidecimal (**.hex**) or Tabular Text File (**.ttf**) format, will have different file sizes. However, for any specific version of the Quartus® II software, any design targeted for the same device will have the same uncompressed configuration file size. If you are using compression, the file size can vary after each compilation because the compression ratio is dependent on the design.

This chapter explains the Stratix II and Stratix II GX device configuration features and describes how to configure Stratix II and Stratix II GX devices using the supported configuration schemes. This chapter provides configuration pin descriptions and the Stratix II and Stratix II GX device configuration file formats. In this chapter, the generic term device(s) includes all Stratix II and Stratix II GX devices.

For more information on setting device configuration options or creating configuration files, refer to *Software Settings* in volume 2 of the *Configuration Handbook*.

Configuration Features

Stratix II and Stratix II GX devices offer configuration data decompression to reduce configuration file storage, design security using data encryption to protect your designs, and remote system upgrades to allow for remotely updating your Stratix II and Stratix II GX designs. Table 7–4 summarizes which configuration features can be used in each configuration scheme.

Table 7–4. Stratix II and Stratix II GX Configuration Features

Configuration Scheme	Configuration Method	Design Security	Decompression	Remote System Upgrade
FPP	MAX II device or a Microprocessor with flash memory	✓ (1)	✓ (1)	✓
	Enhanced Configuration Device		✓ (2)	✓
AS	Serial Configuration Device	✓	✓	✓ (3)
PS	MAX II device or a Microprocessor with flash memory	✓	✓	✓
	Enhanced Configuration Device	✓	✓	✓
	Download cable	✓	✓	
PPA	MAX II device or a Microprocessor with flash memory			✓
JTAG	MAX II device or a Microprocessor with flash memory			

Notes to Table 7–4:

(1) In these modes, the host system must send a DCLK that is 4× the data rate.

(2) The enhanced configuration device decompression feature is available, while the Stratix II and Stratix II GX decompression feature is not available.

(3) Only remote update mode is supported when using the AS configuration scheme. Local update mode is not supported.

Configuration Data Decompression

Stratix II and Stratix II GX devices support configuration data decompression, which saves configuration memory space and time. This feature allows you to store compressed configuration data in configuration devices or other memory and transmit this compressed bit stream to Stratix II and Stratix II GX devices. During configuration, Stratix II and Stratix II GX devices automatically recognize the compressed file format and decompresses the bit stream in real time and programs its SRAM cells.

Data indicates that compression typically reduces configuration bit stream size by 35 to 55%.

Stratix II and Stratix II GX devices support decompression in the FPP (when using a MAX II device/microprocessor + flash), AS, and PS configuration schemes. Decompression is not supported in the PPA configuration scheme nor in JTAG-based configuration.

When using FPP mode, the intelligent host must provide a DCLK that is 4× the data rate. Therefore, the configuration data must be valid for four DCLK cycles.

The decompression feature supported by Stratix II and Stratix II GX devices is different from the decompression feature in enhanced configuration devices (EPC16, EPC8, and EPC4 devices), although they both use the same compression algorithm. The data decompression feature in the enhanced configuration devices allows them to store compressed data and decompress the bitstream before transmitting it to the target devices. When using Stratix II and Stratix II GX devices in FPP mode with enhanced configuration devices, the decompression feature is available only in the enhanced configuration device, not the Stratix II or Stratix II GX device.

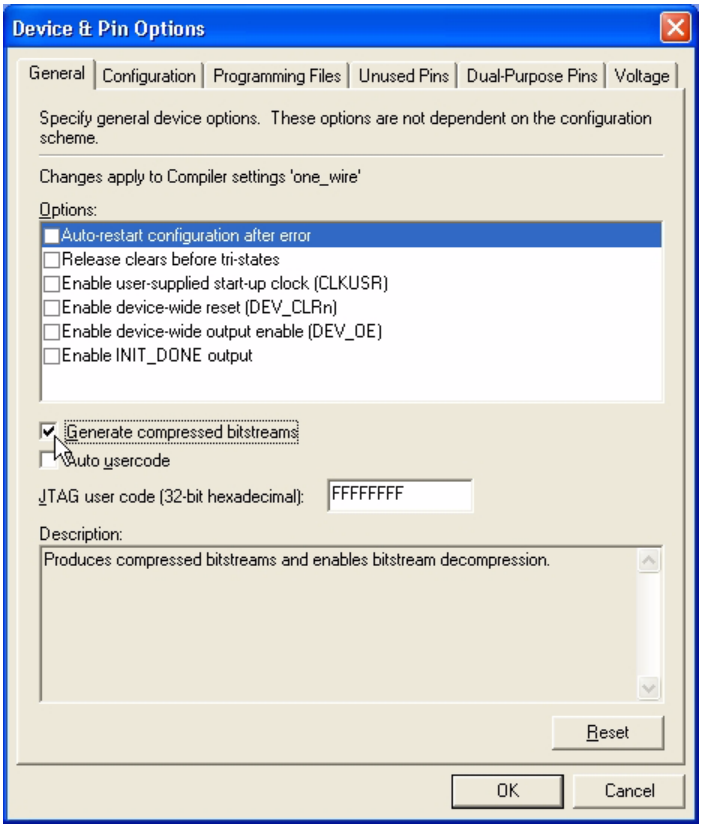
In PS mode, use the Stratix II or Stratix II GX decompression feature because sending compressed configuration data reduces configuration time. Do not use both the Stratix II or Stratix II GX device and the enhanced configuration device decompression features simultaneously. The compression algorithm is not intended to be recursive and could expand the configuration file instead of compressing it further.

When you enable compression, the Quartus II software generates configuration files with compressed configuration data. This compressed file reduces the storage requirements in the configuration device or flash memory, and decreases the time needed to transmit the bitstream to the Stratix II or Stratix II GX device. The time required by a Stratix II or Stratix II GX device to decompress a configuration file is less than the time needed to transmit the configuration data to the device.

There are two ways to enable compression for Stratix II and Stratix II GX bitstreams: before design compilation (in the **Compiler Settings** menu) and after design compilation (in the **Convert Programming Files** window).

To enable compression in the project's compiler settings, select **Device** under the **Assignments** menu to bring up the **Settings** window. After selecting your Stratix II or Stratix II GX device, open the **Device & Pin Options** window, and in the **General** settings tab enable the check box for **Generate compressed bitstreams** (as shown in Figure 7–1).

Figure 7–1. Enabling Compression for Stratix II and Stratix II GX Bitstreams in Compiler Settings



Compression can also be enabled when creating programming files from the **Convert Programming Files** window.

1. Click **Convert Programming Files** (File menu).
2. Select the programming file type (POF, SRAM HEXOUT, RBF, or TTF).
3. For POF output files, select a configuration device.
4. In the **Input files to convert** box, select **SOF Data**.
5. Select **Add File** and add a Stratix II or Stratix II GX device SOF(s).

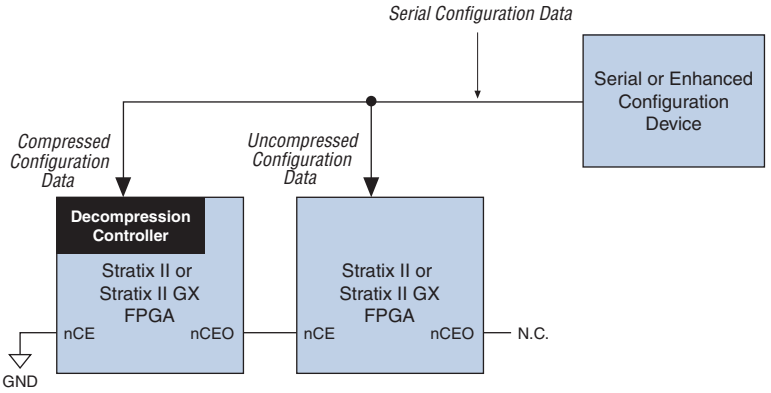
6. Select the name of the file you added to the **SOF Data** area and click **Properties**.

7. Check the **Compression** check box.

When multiple Stratix II or Stratix II GX devices are cascaded, you can selectively enable the compression feature for each device in the chain if you are using a serial configuration scheme. Figure 7–2 depicts a chain of two Stratix II or Stratix II GX devices. The first Stratix II or Stratix II GX device has compression enabled and therefore receives a compressed bit stream from the configuration device. The second Stratix II or Stratix II GX device has the compression feature disabled and receives uncompressed data.

In a multi-device FPP configuration chain all Stratix II or Stratix II GX devices in the chain must either enable of disable the decompression feature. You can not selectively enable the compression feature for each device in the chain because of the DATA and DCLK relationship.

Figure 7–2. Compressed and Uncompressed Configuration Data in the Same Configuration File



You can generate programming files for this setup from the **Convert Programming Files** window (File menu) in the Quartus II software.

Design Security Using Configuration Bitstream Encryption

Stratix II and Stratix II GX devices are the industry's first devices with the ability to decrypt a configuration bitstream using the Advanced Encryption Standard (AES) algorithm—the most advanced encryption algorithm available today. When using the design security feature, a

128-bit security key is stored in the Stratix II or Stratix II GX device. In order to successfully configure a Stratix II or Stratix II GX device that has the design security feature enabled, it must be configured with a configuration file that was encrypted using the same 128-bit security key. The security key can be stored in non-volatile memory inside the Stratix II or Stratix II GX device. This non-volatile memory does not require any external devices, such as a battery back-up, for storage.

When using a serial configuration scheme such as passive serial (PS) or active serial (AS), configuration time is the same whether or not the design security feature is enabled. If the fast passive parallel (FPP) scheme is used with the design security or decompression feature, a 4× DCLK is required. This results in a slower configuration time when compared to the configuration time of an FPGA that has neither the design security, nor decompression feature enabled. For more information about this feature, contact Altera Applications group.

Remote System Upgrade

Stratix II and Stratix II GX devices feature remote and local update.

For more information about this feature, refer to the *Remote System Upgrades With Stratix II & Stratix II GX Devices* chapter in volume 2 of the *Stratix II Device Handbook* or the *Remote System Upgrades With Stratix II & Stratix II GX Devices* chapter in volume 2 of the *Stratix II GX Device Handbook*

Power-On Reset Circuit

The POR circuit keeps the entire system in reset until the power supply voltage levels have stabilized on power-up. Upon power-up, the device does not release nSTATUS until V_{CCINT}, V_{CCPD}, and V_{CCIO} of banks 3, 4, 7, and 8 are above the device's POR trip point. On power down, V_{CCINT} is monitored for brown-out conditions.

The passive serial (PS) mode (MSEL[3,2,1,0] = 0010) and the Fast passive parallel (FPP) mode (MSEL[3,2,1,0] = 0000) always set bank 3 to use the lower POR trip point consistent with 1.8- and 1.5-V signaling, regardless of the VCCSEL setting. For all other configuration modes, VCCSEL selects the POR trip-point level. Refer to the section "VCCSEL Pin" on page 7–10 for more details.

In Stratix II devices, a pin-selectable option PORSEL is provided that allows you to select between a typical POR time setting of 12 ms or 100 ms. In both cases, you can extend the POR time by using an external component to assert the nSTATUS pin low.

V_{CCPD} Pins

Stratix II and Stratix II GX devices also offer a new power supply, V_{CCPD}, which must be connected to 3.3-V in order to power the 3.3-V/2.5-V buffer available on the configuration input pins and JTAG pins. V_{CCPD} applies to all the JTAG input pins (TCK, TMS, TDI, and TRST) and the configuration pins when VCCSEL is connected to ground. Refer to Table 7–5 for information on the pins affected by VCCSEL.

V_{CCPD} must ramp-up from 0-V to 3.3-V within 100 ms. If V_{CCPD} is not ramped up within this specified time, your Stratix II or Stratix II GX device will not configure successfully. If your system does not allow for a V_{CCPD} ramp-up time of 100 ms or less, you must hold nCONFIG low until all power supplies are stable.

VCCSEL Pin

The VCCSEL pin selects the type of input buffer used on configuration input pins and it selects the POR trip point voltage level for V_{CCIO} bank 3 powered by VCCIO3 pins.

For more information, refer to Table 7–24 on page 7–105.

The configuration input pins and the PLL_ENA pin (Table 7–5) have a dual buffer design. These pins have a 3.3-V/2.5-V input buffer and a 1.8-V/1.5-V input buffer. The VCCSEL input pin selects which input buffer is used during configuration. The 3.3-V/2.5-V input buffer is powered by V_{CCPD}, while the 1.8-V/1.5-V input buffer is powered by V_{CCIO}. After configuration, the dual-purpose configuration pins are powered by the V_{CCIO} pins of the bank in which they reside. Table 7–5 shows the pins affected by VCCSEL.

Table 7–5. Pins Affected by the Voltage Level at VCCSEL

Pin	VCCSEL = LOW (connected to GND)	VCCSEL = HIGH (connected to V_{CCPD})
nSTATUS (when used as an input)	3.3/2.5-V input buffer is selected. Input buffer is powered by V_{CCPD}.	1.8/1.5-V input buffer is selected. Input buffer is powered by V_{CCIO} of the I/O bank. These input buffers are 3.3 V tolerant.
nCONFIG		
CONF_DONE (when used as an input)		
DATA[7..0]		
nCE		
DCLK (when used as an input)		
CS		
nWS		
nRS		
nCS		
CLKUSR		
DEV_OE		
DEV_CLRn		
RUnLU		
PLL_ENA		

VCCSEL is sampled during power-up. Therefore, the VCCSEL setting cannot change on the fly or during a reconfiguration. The VCCSEL input buffer is powered by V_{CCINT} and has an internal 5-kΩ pull-down resistor that is always active.

VCCSEL must be hardwired to V_{CCPD} or GND.

A logic high selects the 1.8-V/1.5-V input buffer, and a logic low selects the 3.3-V/2.5-V input buffer. VCCSEL should be set to comply with the logic levels driven out of the configuration device or MAX II device or a microprocessor with flash memory.

VCCSEL also sets the POR trip point for I/O bank 3 to ensure that this I/O bank has powered up to the appropriate voltage levels before configuration begins. For passive serial (PS) mode (MSEL[3..0] = 0010) and for Fast passive parallel (FPP) mode (MSEL[3..0] = 0000) the POR circuitry selects the trip point associated with 1.5-V/1.8-V signaling. For all other configuration modes defined by MSEL[3..0] settings (other

than 00X0 (MSEL[1] = X, "don't care"), VCCSEL=GND selects the higher I/O bank 3 POR trip point for 2.5-V/3.3-V signaling and VCCSEL=VCCPD selects the lower I/O bank 3 POR trip point associated with 1.5-V/1.8-V signaling.

For all configuration modes with MSEL[3..0] not equal to 00X0 (MSEL[1] = X, "don't care"), if VCCIO of configuration bank 3 is powered by 1.8-V or 1.5-V and VCCSEL = GND, the voltage supplied to this I/O bank(s) may never reach the POR trip point, which prevents the device from beginning configuration.

If the VCCIO of I/O bank 3 is powered by 1.5- or 1.8-V and the configuration signals used require 3.3- or 2.5-V signaling, you should set VCCSEL to VCCPD to enable the 1.8-/1.5-V input buffers for configuration. The 1.8-V/1.5-V input buffers are 3.3-V tolerant.

The fast passive parallel (FPP) and passive serial (PS) modes always enable bank 3 to use the POR trip point to be consistent with 1.8- and 1.5-V signaling, regardless of the VCCSEL setting.

Table 7–6 shows how you should set VCCSEL depending on the configuration mode, the voltage level on VCCIO3 pins that power bank 3, and the supported configuration input voltages.

Table 7–6. Supported V_{CCSEL} Setting Based on Mode, VCCIO3, and Input Configuration Voltage

Configuration Mode	V_{CCIO} (Bank 3)	Supported Configuration Input Voltages	V_{CCSEL}
All modes	3.3-V/2.5-V	3.3-V/2.5-V	GND
All modes	1.8-V/1.5-V	3.3-V/2.5-V	V_{CCPD} *(1)*
All modes	1.8-V/1.5-V	1.8-V/1.5-V	V_{CCPD}
—	3.3-V/2.5-V	1.8-V/1.5-V	Not Supported

Note to Table 7–6:

(1) The VCCSEL pin can also be connected to GND for PS (MSEL[3..0]=0010) and FPP (MSEL[3..0]=0000) modes.

Table 7–7 shows the configuration mode support for banks 4, 7, and 8.

Table 7–7. Stratix II Configuration Mode Support for Banks 4, 7 and 8

Configuration Mode	Configuration Voltage/V_{CCIO} Support for Banks 4, 7, and 8		
	3.3/3.3	1.8/1.8	3.3/1.8
	VCCSEL = GND	VCCSEL = VCCPD	VCCSEL = GND
Fast passive parallel	Y	Y	Y
Passive parallel asynchronous	Y	Y	Y
Passive serial	Y	Y	Y
Remote system upgrade FPP	Y	Y	Y
Remote system upgrade PPA	Y	Y	Y
Remote system upgrade PS	Y	Y	Y
Fast AS (40 MHz)	Y	Y	Y
Remote system upgrade fast AS (40 MHz)	Y	Y	Y
FPP with decompression and/or design security	Y	Y	Y
Remote system upgrade FPP with decompression and/or design security feature enabled	Y	Y	Y
AS (20 MHz)	Y	Y	Y
Remote system upgrade AS (20 MHz)	Y	Y	Y

Output Configuration Pins

You must verify that the configuration output pins for your chosen configuration modes meet the V_{IH} of the configuration device. Refer to Table 7–22 on page 7–94 for a consolidated list of configuration output pins.

The V_{IH} of 3.3 V or 2.5 V configuration devices will not be met when the V_{CCIO} of the output configuration pins are 1.8 V or 1.5 V. Level shifters will be required to meet the input high level voltage threshold V_{IH}.

Note that AS mode is only applicable for 3.3-V configurations. If I/O bank 3 is less than 3.3 V, level shifters are required on the output pins (DCLK, nCSO, ASDO) from the Stratix II or Stratix II GX device back to the EPCS device.

The key is to ensure the VCCIO voltage of bank 3 is high enough to trip the VCCIO3 POR trip point on power-up. Also, to make sure the configuration device meets the V_{IH} for the configuration input pins based on the selected input buffer.

Fast Passive Parallel Configuration

Fast passive parallel (FPP) configuration in Stratix II and Stratix II GX devices is designed to meet the continuously increasing demand for faster configuration times. Stratix II and Stratix II GX devices are designed with the capability of receiving byte-wide configuration data per clock cycle. Table 7–8 shows the MSEL pin settings when using the FFP configuration scheme.

Table 7–8. Stratix II and Stratix II GX MSEL Pin Settings for FPP Configuration Schemes Notes (1), (2), and (3)

Configuration Scheme	MSEL3	MSEL2	MSEL1	MSEL0
FPP when not using remote system upgrade or decompression and/or design security feature	0	0	0	0
FPP when using remote system upgrade *(4)*	0	1	0	0
FPP with decompression and/or design security feature enabled *(5)*	1	0	1	1
FPP when using remote system upgrade and decompression and/or design security feature *(4)*, *(5)*	1	1	0	0

Notes to Table 7–8:

(1) You must verify the configuration output pins for your chosen configuraiton modes meet the V_{IH} of the configuration device. Refer to Table 7–22 for a consolidated list of configuration output pins.

(2) The V_{IH} of 3.3-V or 2.5-V configuration devices will not be met when the VCCIO of the output configuration pins is 1.8-V or 1.5-V. Level shifters will be required to meet the input high level voltage threshold V_{IH}.

(3) The VCCSEL signal does not control TDO or nCEO. During configuration, these pins drive out voltage levels corresponding to the VCCIO supply voltage that powers the I/O bank containing the pin. For more information about multi-volt support, including information about using TDO and nCEO in multi-volt systems, refer to the *Stratix II GX Architecture* chapter in volume 1 of the *Stratix II GX Device Handbook*.

(4) These schemes require that you drive the RUnLU pin to specify either remote update or local update. For more information about remote system upgrade in Stratix II devices, refer to the *Remote System Upgrades With Stratix II & Stratix II GX Devices* chapter in volume 2 of the *Stratix II Device Handbook* or the *Remote System Upgrades With Stratix II & Stratix II GX Devices* chapter in volume 2 of the *Stratix II GX Device Handbook*.

(5) These modes are only supported when using a MAX II device or a microprocessor with flash memory for configuration. In these modes, the host system must output a DCLK that is 4× the data rate.

FPP configuration of Stratix II and Stratix II GX devices can be performed using an intelligent host, such as a MAX II device, a microprocessor, or an Altera enhanced configuration device.

FPP Configuration Using a MAX II Device as an External Host

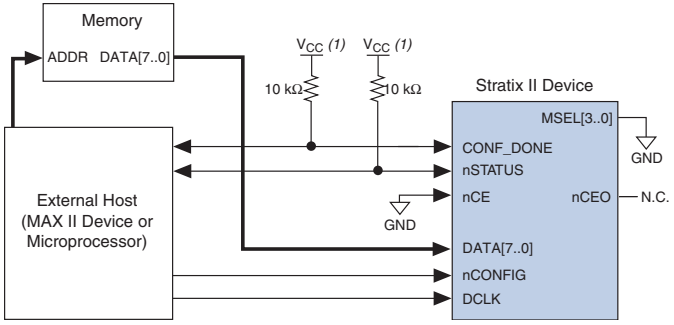
FPP configuration using compression and an external host provides the fastest method to configure Stratix II and Stratix II GX devices. In the FPP configuration scheme, a MAX II device can be used as an intelligent host that controls the transfer of configuration data from a storage device, such as flash memory, to the target Stratix II or Stratix II GX device. Configuration data can be stored in RBF, HEX, or TTF format. When using the MAX II devices as an intelligent host, a design that controls the configuration process, such as fetching the data from flash memory and sending it to the device, must be stored in the MAX II device.

If you are using the Stratix II or Stratix II GX decompression and/or design security feature, the external host must be able to send a DCLK frequency that is 4× the data rate.

The 4× DCLK signal does not require an additional pin and is sent on the DCLK pin. The maximum DCLK frequency is 100 MHz, which results in a maximum data rate of 200 Mbps. If you are not using the Stratix II or Stratix II GX decompression or design security features, the data rate is 8× the DCLK frequency.

Figure 7–3 shows the configuration interface connections between the Stratix II or Stratix II GX device and a MAX II device for single device configuration.

Figure 7–3. Single Device FPP Configuration Using an External Host



Note to Figure 7–3:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for the device. V_{CC} should be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

Upon power-up, the Stratix II and Stratix II GX devices go through a Power-On Reset (POR). The POR delay is dependent on the PORSEL pin setting; when PORSEL is driven low, the POR time is approximately 100 ms, if PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device resets, holds nSTATUS low, and tri-states all user I/O pins. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins have weak pull-up resistors, which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

You can hold nConfig low in order to stop device configuration.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *DC & Switching Characteristics* chapter in volume 1 of the *Stratix II Device Handbook* or the *DC & Switching Characteristics* chapter in volume 1 of the *Stratix II GX Device Handbook*.

The configuration cycle consists of three stages: reset, configuration and initialization. While nCONFIG or nSTATUS are low, the device is in the reset stage. To initiate configuration, the MAX II device must drive the nCONFIG pin from low-to-high.

V_{CCINT}, V_{CCIO}, and V_{CCPD} of the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. Once nSTATUS is released, the device is ready to receive configuration data and the configuration stage begins. When nSTATUS is pulled high, the MAX II device places the configuration data one byte at a time on the DATA[7..0] pins.

Stratix II and Stratix II GX devices receive configuration data on the DATA[7..0] pins and the clock is received on the DCLK pin. Data is latched into the device on the rising edge of DCLK. If you are using the Stratix II or Stratix II GX decompression and/or design security feature, configuration data is latched on the rising edge of every fourth DCLK cycle. After the configuration data is latched in, it is processed during the following three DCLK cycles.

Data is continuously clocked into the target device until CONF_DONE goes high. The CONF_DONE pin goes high one byte early in parallel configuration (FPP and PPA) modes. The last byte is required for serial configuration (AS and PS) modes. After the device has received the next to last byte of the configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by an external 10-kΩ pull-up resistor. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

In Stratix II and Stratix II GX devices, the initialization clock source is either the internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Stratix II or Stratix II GX device provides itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device. Driving DCLK to the device after configuration is complete does not affect device operation.

You can also synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. The **Enable user-supplied start-up clock** (**CLKUSR**) option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR does not affect the configuration process. The CONF_DONE pin goes high one byte early in parallel configuration (FPP and PPA) modes. The last byte is required for serial configuration (AS and PS) modes. After the CONF_DONE pin transitions high, CLKUSR is enabled after the time specified as t_{CD2CU}. After this time period elapses, Stratix II and Stratix II GX devices require 299 clock cycles to initialize properly and enter user mode. Stratix II and Stratix II GX devices support a CLKUSR f_{MAX} of 100 MHz.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. This **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used, it is high because of an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. The MAX II device must be able to detect this low-to-high transition, which signals the device has

entered user mode. When initialization is complete, the device enters user mode. In user-mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

To ensure DCLK and DATA[7..0] are not left floating at the end of configuration, the MAX II device must drive them either high or low, whichever is convenient on your board. The DATA[7..0] pins are available as user I/O pins after configuration. When you select the FPP scheme in the Quartus II software, as a default, these I/O pins are tri-stated in user mode. To change this default option in the Quartus II software, select the **Pins** tab of the **Device & Pin Options** dialog box.

The configuration clock (DCLK) speed must be below the specified frequency to ensure correct configuration. No maximum DCLK period exists, which means you can pause configuration by halting DCLK for an indefinite amount of time.

If you are using the Stratix II or Stratix II GX decompression and/or design security feature and need to stop DCLK, it can only be stopped three clock cycles after the last data byte was latched into the Stratix II or Stratix II GX device.

By stopping DCLK, the configuration circuit allows enough clock cycles to process the last byte of latched configuration data. When the clock restarts, the MAX II device must provide data on the DATA[7..0] pins prior to sending the first DCLK rising edge.

If an error occurs during configuration, the device drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the MAX II device that there is an error. If the **Auto-restart configuration after error** option (available in the Quartus II software from the **General** tab of the **Device & Pin Options** (dialog box) is turned on, the device releases nSTATUS after a reset time-out period (maximum of 100 µs). After nSTATUS is released and pulled high by a pull-up resistor, the MAX II device can try to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

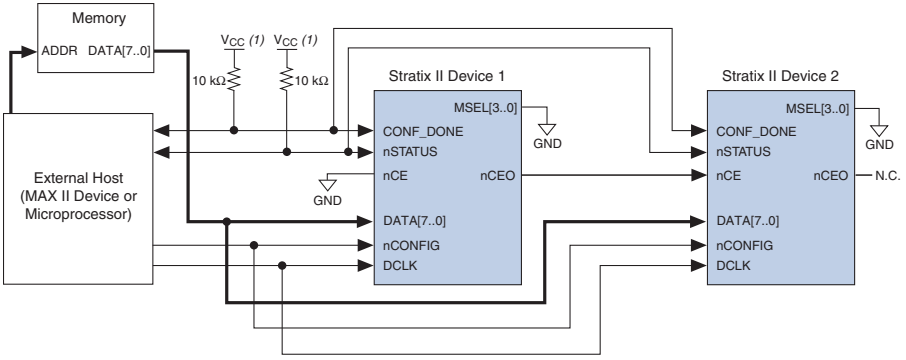
The MAX II device can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The CONF_DONE pin must be monitored by the MAX II device to detect errors and determine when programming completes. If all configuration data is sent, but the CONF_DONE or INIT_DONE signals have not gone high, the MAX II device will reconfigure the target device.

If the optional CLKUSR pin is used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure CLKUSR continues toggling during the time nSTATUS is low (maximum of 100 µs).

When the device is in user-mode, initiating a reconfiguration is done by transitioning the nCONFIG pin low-to-high. The nCONFIG pin should be low for at least 2 µs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Once nCONFIG returns to a logic high level and nSTATUS is released by the device, reconfiguration begins.

Figure 7–4 shows how to configure multiple devices using a MAX II device. This circuit is similar to the FPP configuration circuit for a single device, except the Stratix II or Stratix II GX devices are cascaded for multi-device configuration.

Figure 7–4. Multi-Device FPP Configuration Using an External Host



Note to Figure 7–4:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain. V_{CC} should be high enough to meet the V_{IH} specification of the I/O standard on the device and the external host.

In multi-device FPP configuration, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration within one clock cycle; therefore, the transfer of data destinations is transparent to the MAX II device. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in

the chain. The configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

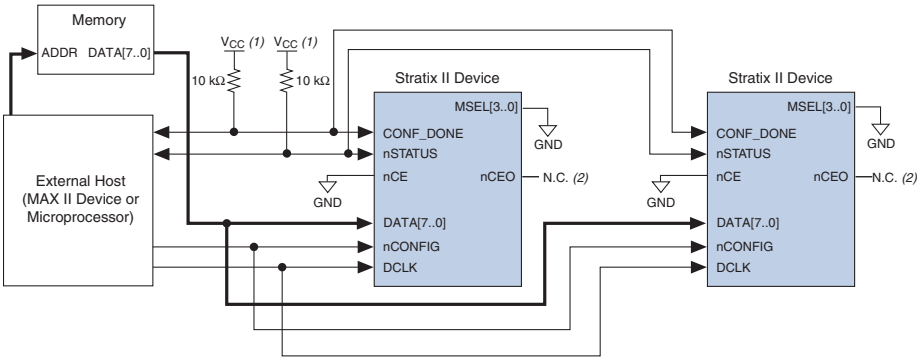
All nSTATUS and CONF_DONE pins are tied together and if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

If the **Auto-restart configuration after error** option is turned on, the devices release their nSTATUS pins after a reset time-out period (maximum of 100 µs). After all nSTATUS pins are released and pulled high, the MAX II device can try to reconfigure the chain without pulsing nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

In a multi-device FPP configuration chain, all Stratix II or Stratix II GX devices in the chain must either enable or disable the decompression and/or design security feature. You can not selectively enable the decompression and/or design security feature for each device in the chain because of the DATA and DCLK relationship. If the chain contains devices that do not support design security, you should use a serial configuration scheme.

If a system has multiple devices that contain the same configuration data, tie all device nCE inputs to GND, and leave nCEO pins floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. Configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices start and complete configuration at the same time. Figure 7–5 shows multi-device FPP configuration when both Stratix II or Stratix II GX devices are receiving the same configuration data.

Figure 7–5. Multiple-Device FPP Configuration Using an External Host When Both Devices Receive the Same Data



Notes to Figure 7–5:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain. V_{CC} should be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

(2) The nCEO pins of both Stratix II or Stratix II GX devices are left unconnected when configuring the same configuration data into multiple devices.

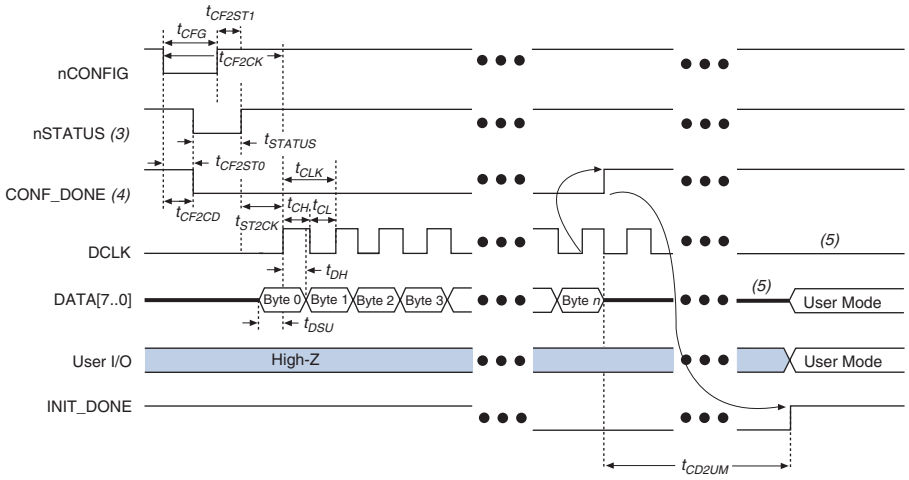
You can use a single configuration chain to configure Stratix II or Stratix II GX devices with other Altera devices that support FPP configuration, such as Stratix devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, tie all of the device CONF_DONE and nSTATUS pins together.

For more information on configuring multiple Altera devices in the same configuration chain, refer to *Configuring Mixed Altera FPGA Chains* in volume 2 of the *Configuration Handbook*.

FPP Configuration Timing

Figure 7–6 shows the timing waveform for FPP configuration when using a MAX II device as an external host. This waveform shows the timing when the decompression and the design security feature are not enabled.

Figure 7–6. FPP Configuration Timing Waveform *Notes (1), (2)*



Notes to Figure 7–6:

(1) This timing waveform should be used when the decompression and design security feature are not used.
(2) The beginning of this waveform shows the device in user-mode. In user-mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.
(3) Upon power-up, the Stratix II or Stratix II GX device holds nSTATUS low for the time of the POR delay.
(4) Upon power-up, before and during configuration, CONF_DONE is low.
(5) DCLK should not be left floating after configuration. It should be driven high or low, whichever is more convenient.
(6) DATA[7..0] are available as user I/O pins after configuration and the state of these pins depends on the dual-purpose pin settings.

Table 7–9 defines the timing parameters for Stratix II and Stratix II GX devices for FPP configuration when the decompression and the design security features are not enabled.

Table 7–9. FPP Timing Parameters for Stratix II and Stratix II GX Devices (Part 1 of 2) *Notes (1), (2)*

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		800	ns
t_{CFG}	nCONFIG low pulse width	2		µs
t_{STATUS}	nSTATUS low pulse width	10	100 *(3)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high		100 *(3)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	100		µs
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2		µs

Table 7–9. FPP Timing Parameters for Stratix II and Stratix II GX Devices (Part 2 of 2) *Notes (1), (2)*

Symbol	Parameter	Min	Max	Units
t_{DSU}	Data setup time before rising edge on DCLK	5		ns
t_{DH}	Data hold time after rising edge on DCLK	0		ns
t_{CH}	DCLK high time	4		ns
t_{CL}	DCLK low time	4		ns
t_{CLK}	DCLK period	10		ns
f_{MAX}	DCLK frequency		100	MHz
t_R	Input rise time		40	ns
t_F	Input fall time		40	ns
t_{CD2UM}	CONF_DONE high to user mode *(4)*	20	100	µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period		
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	t_{CD2CU} +(299 × CLKUSR period)		

Notes to Table 7–9:

(1) This information is preliminary.

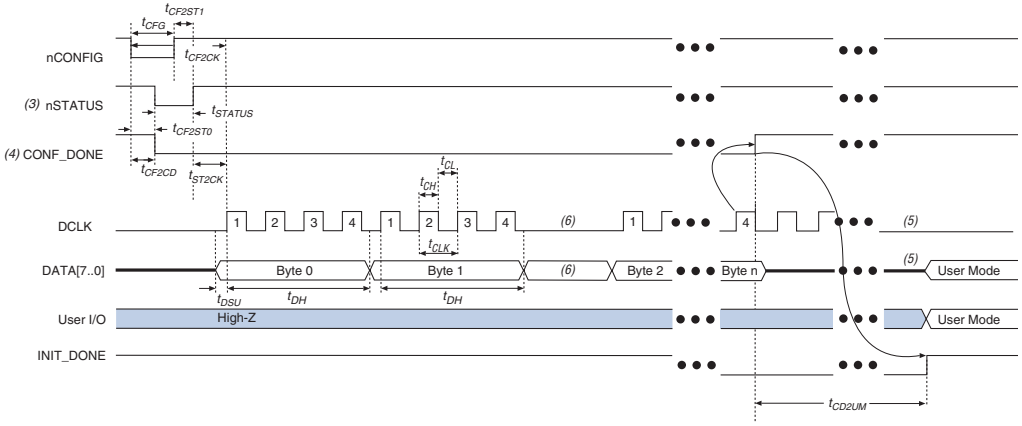
(2) These timing parameters should be used when the decompression and design security feature are not used.

(3) This value is obtainable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(4) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device.

Figure 7–7 shows the timing waveform for FPP configuration when using a MAX II device as an external host. This waveform shows the timing when the decompression and/or the design security feature are enabled.

Figure 7–7. FPP Configuration Timing Waveform With Decompression or Design Security Feature Enabled *Notes (1), (2)*



Notes to Figure 7–7:

(1) This timing waveform should be used when the decompression and/or design security feature are used.
(2) The beginning of this waveform shows the device in user-mode. In user-mode, `nCONFIG`, `nSTATUS` and `CONF_DONE` are at logic high levels. When `nCONFIG` is pulled low, a reconfiguration cycle begins.
(3) Upon power-up, the Stratix II or Stratix II GX device holds `nSTATUS` low for the time of the POR delay.
(4) Upon power-up, before and during configuration, `CONF_DONE` is low.
(5) `DCLK` should not be left floating after configuration. It should be driven high or low, whichever is more convenient.
(6) `DATA[7..0]` are available as user I/O pins after configuration and the state of these pins depends on the dual-purpose pin settings.
(7) If needed, `DCLK` can be paused by holding it low. When `DCLK` restarts, the external host must provide data on the `DATA[7..0]` pins prior to sending the first `DCLK` rising edge.

Table 7–10 defines the timing parameters for Stratix II and Stratix II GX devices for FPP configuration when the decompression and/or the design security feature are enabled.

Table 7–10. FPP Timing Parameters for Stratix II and Stratix II GX Devices With Decompression or Design Security Feature Enabled *Note (1)*

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		800	ns
t_{CFG}	nCONFIG low pulse width	2		µs
t_{STATUS}	nSTATUS low pulse width	10	100 *(2)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high		100 *(2)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	100		µs
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2		µs
t_{DSU}	Data setup time before rising edge on DCLK	5		ns
t_{DH}	Data hold time after rising edge on DCLK	30		ns
t_{CH}	DCLK high time	4		ns
t_{CL}	DCLK low time	4		ns
t_{CLK}	DCLK period	10		ns
f_{MAX}	DCLK frequency		100	MHz
t_{DATA}	Data rate		200	Mbps
t_{R}	Input rise time		40	ns
t_{F}	Input fall time		40	ns
t_{CD2UM}	CONF_DONE high to user mode *(3)*	20	100	µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period		
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	t_{CD2CU} + (299 × CLKUSR period)		

Notes to Table 7–10:

(1) These timing parameters should be used when the decompression and design security feature are used.

(2) This value is obtainable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(3) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device.

Device configuration options and how to create configuration files are discussed further in the *Software Settings* chapter in the *Configuration Handbook*.

FPP Configuration Using a Microprocessor

In the FPP configuration scheme, a microprocessor can control the transfer of configuration data from a storage device, such as flash memory, to the target Stratix II or Stratix II GX device.

All information in "FPP Configuration Using a MAX II Device as an External Host" on page 7–15 is also applicable when using a microprocessor as an external host. Refer to that section for all configuration and timing information.

FPP Configuration Using an Enhanced Configuration Device

In the FPP configuration scheme, an enhanced configuration device sends a byte of configuration data every DCLK cycle to the Stratix II or Stratix II GX device. Configuration data is stored in the configuration device.

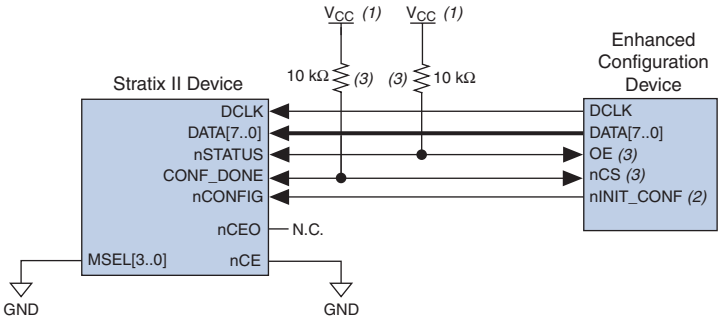
When configuring your Stratix II or Stratix II GX device using FPP mode and an enhanced configuration device, the enhanced configuration device decompression feature is available while the Stratix II and Stratix II GX decompression and design security features are not.

Figure 7–8 shows the configuration interface connections between a Stratix II or Stratix II GX device and the enhanced configuration device for single device configuration.

The figures in this chapter only show the configuration-related pins and the configuration pin connections between the configuration device and the device.

For more information on the enhanced configuration device and flash interface pins, such as PGM[2..0], EXCLK, PORSEL, A[20..0], and DQ[15..0], refer to the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* in volume 2 of the *Configuration Handbook*.

Figure 7–8. Single Device FPP Configuration Using an Enhanced Configuration Device



Notes to Figure 7–8:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active. This means an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

The value of the internal pull-up resistors on the enhanced configuration devices can be found in the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* in volume 2 of the *Configuration Handbook*.

When using enhanced configuration devices, you can connect the device's nCONFIG pin to nINIT_CONF pin of the enhanced configuration device, which allows the INIT_CONF JTAG instruction to initiate device configuration. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor. An internal pull-up resistor on the nINIT_CONF pin is always active in the enhanced configuration devices, which means an external pull-up resistor should not be used if nCONFIG is tied to nINIT_CONF.

Upon power-up, the Stratix II or Stratix II GX device goes through a POR. The POR delay is dependent on the PORSEL pin setting; when PORSEL is driven low, the POR time is approximately 100 ms, if PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device will reset, hold nSTATUS low, and tri-state all user I/O pins. The

configuration device also goes through a POR delay to allow the power supply to stabilize. The POR time for enhanced configuration devices can be set to either 100 ms or 2 ms, depending on its PORSEL pin setting. If the PORSEL pin is connected to GND, the POR delay is 100 ms. If the PORSEL pin is connected to V_{CC}, the POR delay is 2 ms. During this time, the configuration device drives its OE pin low. This low signal delays configuration because the OE pin is connected to the target device's nSTATUS pin.

When selecting a POR time, you need to ensure that the device completes power-up before the enhanced configuration device exits POR. Altera recommends that you use a 12-ms POR time for the Stratix II or Stratix II GX device, and use a 100-ms POR time for the enhanced configuration device.

When both devices complete POR, they release their open-drain OE or nSTATUS pin, which is then pulled high by a pull-up resistor. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors, which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *Stratix II Device Handbook* or the *Stratix II GX Device Handbook*.

When the power supplies have reached the appropriate operating voltages, the target device senses the low-to-high transition on nCONFIG and initiates the configuration cycle. The configuration cycle consists of three stages: reset, configuration and initialization. While nCONFIG or nSTATUS are low, the device is in reset. The beginning of configuration can be delayed by holding the nCONFIG or nSTATUS pin low.

V_{CCINT}, V_{CCIO} and V_{CCPD} of the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the nSTATUS pin, which is pulled high by a pull-up resistor. Enhanced configuration devices have an optional internal pull-up resistor on the OE pin. This option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If this internal pull-up resistor is not used, an external 10-kΩ pull-up resistor on the OE-nSTATUS line is required. Once nSTATUS is released, the device is ready to receive configuration data and the configuration stage begins.

When nSTATUS is pulled high, the configuration device's OE pin also goes high and the configuration device clocks data out to the device using the Stratix II or Stratix II GX device's internal oscillator. The Stratix II and Stratix II GX devices receive configuration data on the DATA[7..0] pins and the clock is received on the DCLK pin. A byte of data is latched into the device on each rising edge of DCLK.

After the device has received all configuration data successfully, it releases the open-drain CONF_DONE pin which is pulled high by a pull-up resistor. Because CONF_DONE is tied to the configuration device's nCS pin, the configuration device is disabled when CONF_DONE goes high. Enhanced configuration devices have an optional internal pull-up resistor on the nCS pin. This option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If this internal pull-up resistor is not used, an external 10-kΩ pull-up resistor on the nCS-CONF_DONE line is required. A low to high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin.

In Stratix II and Stratix II GX devices, the initialization clock source is either the internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Stratix II or Stratix II GX device provides itself with enough clock cycles for proper initialization. You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. The **Enable user-supplied start-up clock** (**CLKUSR**) option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR will not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, CLKUSR will be enabled after the time specified as t_{CD2CU}. After this time period elapses, Stratix II and Stratix II GX devices require 299 clock cycles to initialize properly and enter user mode. Stratix II and Stratix II GX devices support a CLKUSR f_{MAX} of 100 MHz.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used, it will be high due to an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low. When initialization is complete, the INIT_DONE pin will be released and pulled high. In user-mode, the user

I/O pins will no longer have weak pull-up resistors and will function as assigned in your design. The enhanced configuration device will drive DCLK low and DATA[7..0] high at the end of configuration.

If an error occurs during configuration, the device drives its nSTATUS pin low, resetting itself internally. Since the nSTATUS pin is tied to OE, the configuration device will also be reset. If the **Auto-restart configuration after error** option (available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box) is turned on, the device will automatically initiate reconfiguration if an error occurs. The Stratix II or Stratix II GX device releases its nSTATUS pin after a reset time-out period (maximum of 100 µs). When the nSTATUS pin is released and pulled high by a pull-up resistor, the configuration device reconfigures the chain. If this option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 2 µs to restart configuration. The external system can pulse nCONFIG if nCONFIG is under system control rather than tied to V_{CC}.

In addition, if the configuration device sends all of its data and then detects that CONF_DONE has not gone high, it recognizes that the device has not configured successfully. Enhanced configuration devices wait for 64 DCLK cycles after the last configuration bit was sent for CONF_DONE to reach a high state. In this case, the configuration device pulls its OE pin low, which in turn drives the target device's nSTATUS pin low. If the Auto-restart configuration after error option is set in the software, the target device resets and then releases its nSTATUS pin after a reset time-out period (maximum of 100 µs). When nSTATUS returns to a logic high level, the configuration device will try to reconfigure the device.

When CONF_DONE is sensed low after configuration, the configuration device recognizes that the target device has not configured successfully. Therefore, your system should not pull CONF_DONE low to delay initialization. Instead, you should use the CLKUSR option to synchronize the initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain will initialize together if their CONF_DONE pins are tied together.

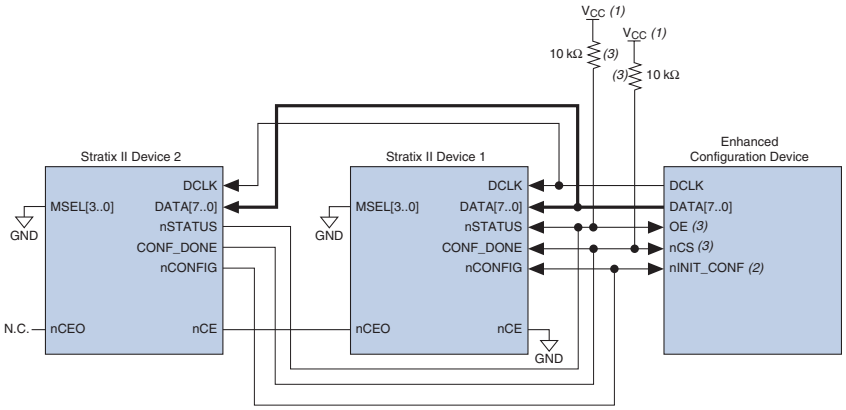
If the optional CLKUSR pin is used and nCONFIG is pulled low to restart configuration during device initialization, ensure CLKUSR continues toggling during the time nSTATUS is low (maximum of 100 µs).

When the device is in user-mode, a reconfiguration can be initiated by pulling the nCONFIG pin low. The nCONFIG pin should be low for at least 2 µs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Because CONF_DONE is

pulled low, this activates the configuration device because it sees its nCS pin drive low. Once nCONFIG returns to a logic high level and nSTATUS is released by the device, reconfiguration begins.

Figure 7–9 shows how to configure multiple Stratix II or Stratix II GX devices with an enhanced configuration device. This circuit is similar to the configuration device circuit for a single device, except the Stratix II or Stratix II GX devices are cascaded for multi-device configuration.

Figure 7–9. Multi-Device FPP Configuration Using an Enhanced Configuration Device



Notes to Figure 7–9:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active. This means an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-up resistors on configuration device** option when generating programming files.

Enhanced configuration devices cannot be cascaded.

When performing multi-device configuration, you must generate the configuration device's POF from each project's SOF. You can combine multiple SOFs using the **Convert Programming Files** window in the Quartus II software.

For more information on how to create configuration files for multi-device configuration chains, refer to *Software Settings* in volume 2 of the *Configuration Handbook*.

In multi-device FPP configuration, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. Pay special attention to the configuration signals because they may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device.

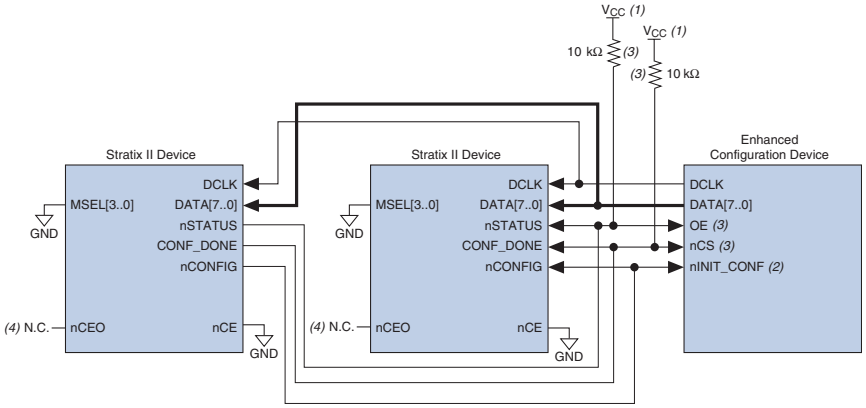
When configuring multiple devices, configuration does not begin until all devices release their OE or nSTATUS pins. Similarly, since all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Since all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This low signal drives the OE pin low on the enhanced configuration device and drives nSTATUS low on all devices, which causes them to enter a reset state. This behavior is similar to a single device detecting an error.

If the **Auto-restart configuration after error** option is turned on, the devices will automatically initiate reconfiguration if an error occurs. The devices will release their nSTATUS pins after a reset time-out period (maximum of 100 µs). When all the nSTATUS pins are released and pulled high, the configuration device tries to reconfigure the chain. If the **Auto-restart configuration after error** option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 2 µs to restart configuration. The external system can pulse nCONFIG if nCONFIG is under system control rather than tied to V_{CC}.

Your system may have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. Configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete configuration at the same time. Figure 7–10 shows multi-device FPP configuration when both Stratix II or Stratix II GX devices are receiving the same configuration data.

Figure 7–10. Multiple-Device FPP Configuration Using an Enhanced Configuration Device When Both devices Receive the Same Data



Notes to Figure 7–10:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active. This means an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

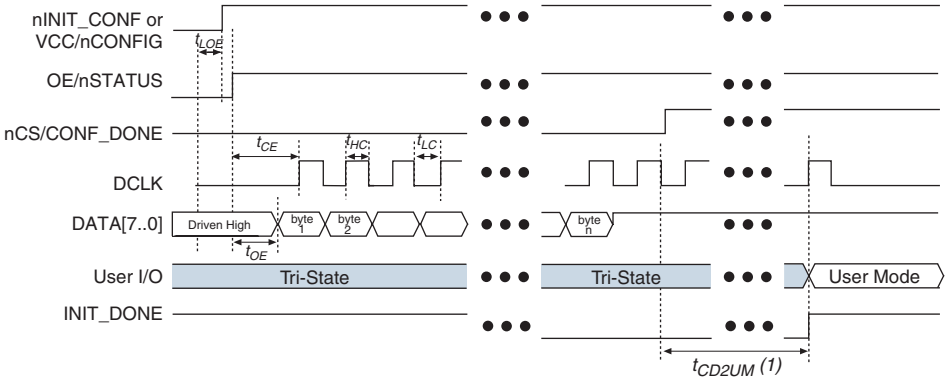
(4) The nCEO pins of both devices are left unconnected when configuring the same configuration data into multiple devices.

You can use a single enhanced configuration chain to configure multiple Stratix II or Stratix II GX devices with other Altera devices that support FPP configuration, such as Stratix and Stratix GX devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information on configuring multiple Altera devices in the same configuration chain, refer to *Configuring Mixed Altera FPGA Chains* in the *Configuration Handbook*.

Figure 7–11 shows the timing waveform for the FPP configuration scheme using an enhanced configuration device.

Figure 7–11. Stratix II and Stratix II GX FPP Configuration Using an Enhanced Configuration Device Timing Waveform



Note to Figure 7–11:
(1) The initialization clock can come from the Stratix II or Stratix II GX device's internal oscillator or the CLKUSR pin.

For timing information, refer to the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* in volume 2 of the *Configuration Handbook*.

Device configuration options and how to create configuration files are discussed further in the *Software Settings* section in volume 2 of the *Configuration Handbook*.

Active Serial Configuration (Serial Configuration Devices)

In the AS configuration scheme, Stratix II and Stratix II GX devices are configured using a serial configuration device. These configuration devices are low-cost devices with non-volatile memory that feature a simple four-pin interface and a small form factor. These features make serial configuration devices an ideal low-cost configuration solution.

Note that AS mode is only applicable for 3.3-V configurations. If I/O bank 3 is less than 3.3 V, level shifters are required on the output pins (DCLK, nCSO, ASDO) from the Stratix II or Stratix II GX device back to the EPCS device.

If VCCIO in bank 3 is set to 1.8 V, an external voltage level translator is needed to meet the V_{IH} of the EPCS device (3.3 V).

For more information on serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook*.

Serial configuration devices provide a serial interface to access configuration data. During device configuration, Stratix II and Stratix II GX devices read configuration data via the serial interface, decompress data if necessary, and configure their SRAM cells. This scheme is referred to as the AS configuration scheme because the device controls the configuration interface. This scheme contrasts with the PS configuration scheme, where the configuration device controls the interface.

The Stratix II and Stratix II GX decompression and design security features are fully available when configuring your Stratix II or Stratix II GX device using AS mode.

Table 7–11 shows the MSEL pin settings when using the AS configuration scheme.

Table 7–11. Stratix II and Stratix II GX MSEL Pin Settings for AS Configuration Schemes *Note (2)*

Configuration Scheme	MSEL3	MSEL2	MSEL1	MSEL0
Fast AS (40 MHz) *(1)*	1	0	0	0
Remote system upgrade fast AS (40 MHz) *(1)*	1	0	0	1
AS (20 MHz) *(1)*	1	1	0	1
Remote system upgrade AS (20 MHz) *(1)*	1	1	1	0

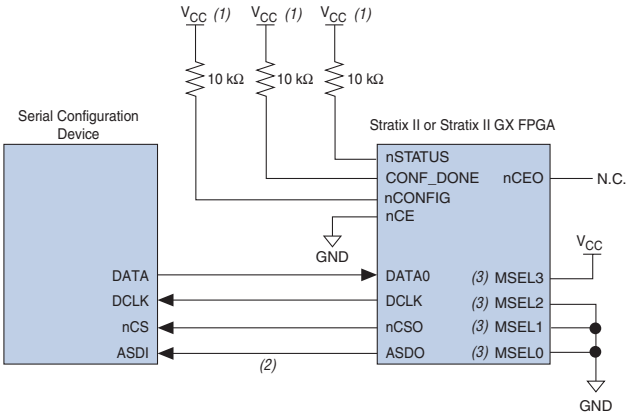
Notes to Table 7–11:

(1) Only the EPCS16 and EPCS64 devices support a DCLK up to 40 MHz clock; other EPCS devices support a DCLK up to 20 MHz. Refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook* for more information.

(2) Note that AS mode is only applicable for 3.3-V configuration. If I/O bank 3 is less than 3.3-V, level shifters are required on the output pins (DCLK,nCSO, and ASDO) from the Stratix II or Stratix II GX device back to the EPCS device.

Serial configuration devices have a four-pin interface: serial clock input (DCLK), serial data output (DATA), AS data input (ASDI), and an active-low chip select (nCS). This four-pin interface connects to Stratix II and Stratix II GX device pins, as shown in Figure 7–12.

Figure 7–12. Single Device AS Configuration



Notes to Figure 7–12:

(1) Connect the pull-up resistors to a 3.3-V supply.

(2) Stratix II and Stratix II GX devices use the ASDO to ASDI path to control the configuration device.

(3) If using an EPCS4 device, MSEL[3..0] should be set to 1101. Refer to Table 7–11 for more details.

Upon power-up, the Stratix II and Stratix II GX devices go through a POR. The POR delay is dependent on the PORSEL pin setting. When PORSEL is driven low, the POR time is approximately 100 ms. If PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device will reset, hold nSTATUS and CONF_DONE low, and tri-state all user I/O pins. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *DC & Switching Characteristics* chapter in volume 1 of the *Stratix II Device Handbook* and the *DC & Switching Characteristics* chapter in volume 1 of the *Stratix II GX Device Handbook*.

The configuration cycle consists of three stages: reset, configuration and initialization. While nCONFIG or nSTATUS are low, the device is in reset. After POR, the Stratix II and Stratix II GX devices release nSTATUS, which is pulled high by an external 10-kΩ pull-up resistor, and enters configuration mode.

To begin configuration, power the V_{CCINT}, V_{CCIO}, and V_{CCPD} voltages (for the banks where the configuration and JTAG pins reside) to the appropriate voltage levels.

The serial clock (DCLK) generated by the Stratix II and Stratix II GX devices controls the entire configuration cycle and provides the timing for the serial interface. Stratix II and Stratix II GX devices use an internal oscillator to generate DCLK. Using the MSEL[] pins, you can select to use either a 40- or 20-MHz oscillator.

Only the EPCS16 and EPCS64 devices support a DCLK up to 40-MHz clock; other EPCS devices support a DCLK up to 20-MHz. Refer to the *Serial Configuration Devices Data Sheet* for more information. The EPCS4 device only supports the smallest Stratix II (EP2S15) device, which is when the SOF compression is enabled. Because of its insufficient memory capacity, the EPCS1 device does not support any Stratix II devices.

Table 7–12 shows the active serial DCLK output frequencies.

Table 7–12. Active Serial DCLK Output Frequency

Oscillator	Minimum	Typical	Maximum	Units
40 MHz *(1)*	20	26	40	MHz
20 MHz	10	13	20	MHz

Note to Table 7–12:

(1) Only the EPCS16 and EPCS64 devices support a DCLK up to 40-MHz clock; other EPCS devices support a DCLK up to 20-MHz. Refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS16, and EPCS128) Data Sheet* chapter in volume 2 of the *Configuration Handbook* for more information.

In both AS and fast AS configuration schemes, the serial configuration device latches input and control signals on the rising edge of DCLK and drives out configuration data on the falling edge. Stratix II and Stratix II GX devices drive out control signals on the falling edge of DCLK and latch configuration data on the falling edge of DCLK.

In configuration mode, Stratix II and Stratix II GX devices enable the serial configuration device by driving the nCSO output pin low, which connects to the chip select (nCS) pin of the configuration device. The Stratix II and Stratix II GX devices use the serial clock (DCLK) and serial data output (ASDO) pins to send operation commands and/or read address signals to the serial configuration device. The configuration device provides data on its serial data output (DATA) pin, which connects to the DATA0 input of the Stratix II and Stratix II GX devices.

After all configuration bits are received by the Stratix II or Stratix II GX device, it releases the open-drain CONF_DONE pin, which is pulled high by an external 10-kΩ resistor. Initialization begins only after the CONF_DONE signal reaches a logic high level. All AS configuration pins, DATA0, DCLK, nCSO, and ASDO, have weak internal pull-up resistors that are always active. After configuration, these pins are set as input tri-stated and are driven high by the weak internal pull-up resistors. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

In Stratix II and Stratix II GX devices, the initialization clock source is either the 10-MHz (typical) internal oscillator (separate from the active serial internal oscillator) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Stratix II or Stratix II GX device provides itself with enough clock cycles for proper initialization. You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. The **Enable user-supplied start-up clock** (**CLKUSR**) option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. When you **Enable** the **user supplied start-up clock** option, the CLKUSR pin is the initialization clock source. Supplying a clock on CLKUSR will not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, CLKUSR is enabled after 600 ns. After this time period elapses, Stratix II and Stratix II GX devices require 299 clock cycles to initialize properly and enter user mode. Stratix II and Stratix II GX devices support a CLKUSR f_{MAX} of 100 MHz.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used, it will be high due to an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. This low-to-high transition signals that the device has entered user mode. When initialization is complete, the device enters user mode. In user mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

If an error occurs during configuration, Stratix II and Stratix II GX devices assert the nSTATUS signal low, indicating a data frame error, and the CONF_DONE signal stays low. If the **Auto-restart configuration after error** option (available in the Quartus II software from the **General** tab of the

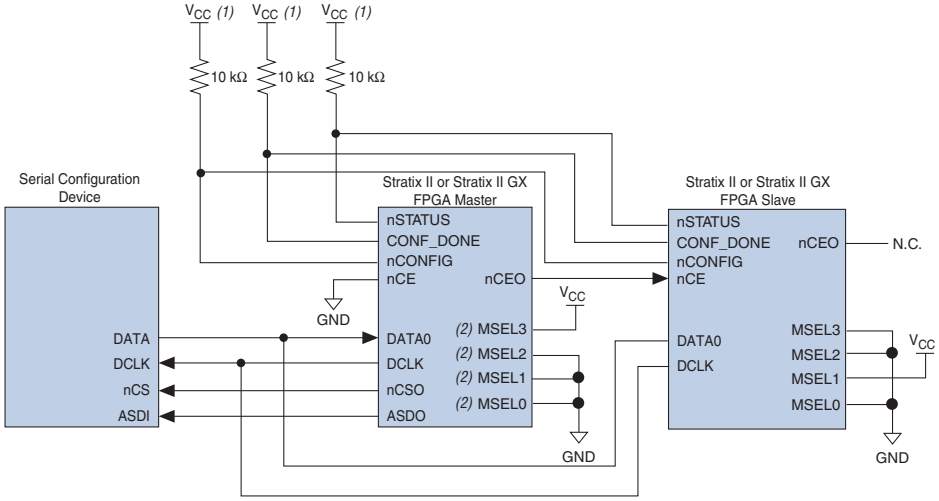
Device & Pin Options dialog box) is turned on, the Stratix II or Stratix II GX device resets the configuration device by pulsing nCSO, releases nSTATUS after a reset time-out period (maximum of 100 µs), and retries configuration. If this option is turned off, the system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 2 µs to restart configuration.

When the Stratix II or Stratix II GX device is in user mode, you can initiate reconfiguration by pulling the nCONFIG pin low. The nCONFIG pin should be low for at least 2 µs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Once nCONFIG returns to a logic high level and nSTATUS is released by the Stratix II or Stratix II GX device, reconfiguration begins.

You can configure multiple Stratix II or Stratix II GX devices using a single serial configuration device. You can cascade multiple Stratix II or Stratix II GX devices using the chip-enable (nCE) and chip-enable-out (nCEO) pins. The first device in the chain must have its nCE pin connected to ground. You must connect its nCEO pin to the nCE pin of the next device in the chain. When the first device captures all of its configuration data from the bit stream, it drives the nCEO pin low, enabling the next device in the chain. You must leave the nCEO pin of the last device unconnected. The nCONFIG, nSTATUS, CONF_DONE, DCLK, and DATA0 pins of each device in the chain are connected (refer to Figure 7–13).

This first Stratix II or Stratix II GX device in the chain is the configuration master and controls configuration of the entire chain. You must connect its MSEL pins to select the AS configuration scheme. The remaining Stratix II or Stratix II GX devices are configuration slaves and you must connect their MSEL pins to select the PS configuration scheme. Any other Altera device that supports PS configuration can also be part of the chain as a configuration slave. Figure 7–13 shows the pin connections for this setup.

Figure 7–13. Multi-Device AS Configuration



Notes to Figure 7–13:

(1) Connect the pull-up resistors to a 3.3-V supply.

(2) If using an EPCS4 device, `MSEL[3..0]` should be set to 1101. Refer to Table 7–11 for more details.

As shown in Figure 7–13, the nSTATUS and CONF_DONE pins on all target devices are connected together with external pull-up resistors. These pins are open-drain bidirectional pins on the devices. When the first device asserts nCEO (after receiving all of its configuration data), it releases its CONF_DONE pin. But the subsequent devices in the chain keep this shared CONF_DONE line low until they have received their configuration data. When all target devices in the chain have received their configuration data and have released CONF_DONE, the pull-up resistor drives a high level on this line and all devices simultaneously enter initialization mode.

If an error occurs at any point during configuration, the nSTATUS line is driven low by the failing device. If you enable the Auto-restart configuration after error option, reconfiguration of the entire chain begins after a reset time-out period (a maximum of 100 µs). If the **Auto-restart configuration after error** option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low to restart configuration. The external system can pulse nCONFIG if it is under system control rather than tied to V_{CC}.

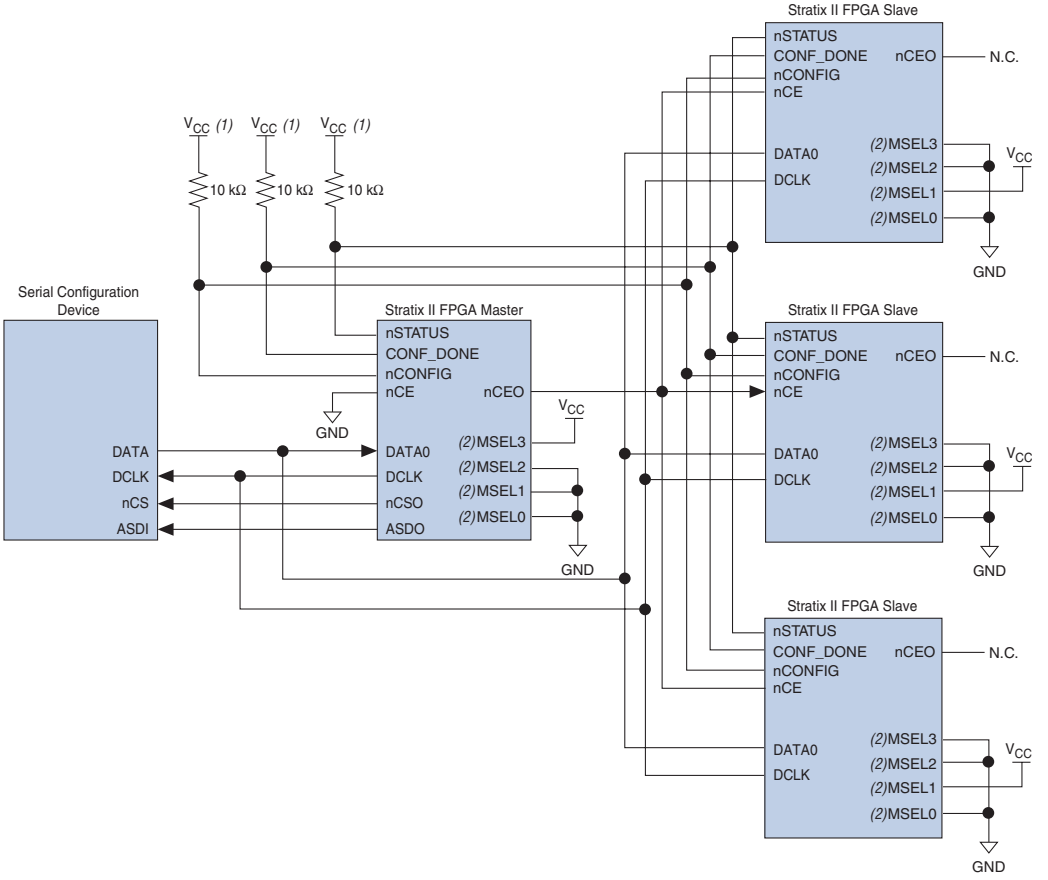
While you can cascade Stratix II or Stratix II GX devices, serial configuration devices cannot be cascaded or chained together.

If the configuration bit stream size exceeds the capacity of a serial configuration device, you must select a larger configuration device and/or enable the compression feature. When configuring multiple devices, the size of the bitstream is the sum of the individual devices' configuration bitstreams.

A system may have multiple devices that contain the same configuration data. In active serial chains, this can be implemented by storing two copies of the SOF in the serial configuration device. The first copy would configure the master Stratix II or Stratix II GX device, and the second copy would configure all remaining slave devices concurrently. All slave devices must be the same density and package. The setup is similar to Figure 7–13, where the master is set up in active serial mode and the slave devices are set up in passive serial mode.

To configure four identical Stratix II or Stratix II GX devices with the same SOF, you could set up the chain similar to the example shown in Figure 7–14. The first device is the master device and its MSEL pins should be set to select AS configuration. The other three slave devices are set up for concurrent configuration and its MSEL pins should be set to select PS configuration. The nCEO pin from the master device drives the nCE input pins on all three slave devices, and the DATA and DCLK pins connect in parallel to all four devices. During the first configuration cycle, the master device reads its configuration data from the serial configuration device while holding nCEO high. After completing its configuration cycle, the master drives nCE low and transmits the second copy of the configuration data to all three slave devices, configuring them simultaneously.

Figure 7–14. Multi-Device AS Configuration When devices Receive the Same Data



Notes to Figure 7–14:

(1) Connect the pull-up resistors to a 3.3-V supply.

(2) If using an EPCS4 device, `MSEL[3..0]` should be set to 1101. Refer to Table 7–11 for more details.

Estimating Active Serial Configuration Time

Active serial configuration time is dominated by the time it takes to transfer data from the serial configuration device to the Stratix II device. This serial interface is clocked by the Stratix II DCLK output (generated from an internal oscillator). As listed in Table 7–12 on page 7–37, the DCLK minimum frequency when choosing to use the 40-MHz oscillator is 20 MHz (50 ns). Therefore, the maximum configuration time estimate for an EP2S15 device (5 MBits of uncompressed data) is:

RBF Size (minimum DCLK period / 1 bit per DCLK cycle) = estimated maximum configuration time

5 Mbits × (50 ns / 1 bit) = 250 ms

To estimate the typical configuration time, use the typical DCLK period as listed in Table 7–12. With a typical DCLK period of 38.46 ns, the typical configuration time is 192 ms. Enabling compression reduces the amount of configuration data that is transmitted to the Stratix II or Stratix II GX device, which also reduces configuration time. On average, compression reduces configuration time by 50%.

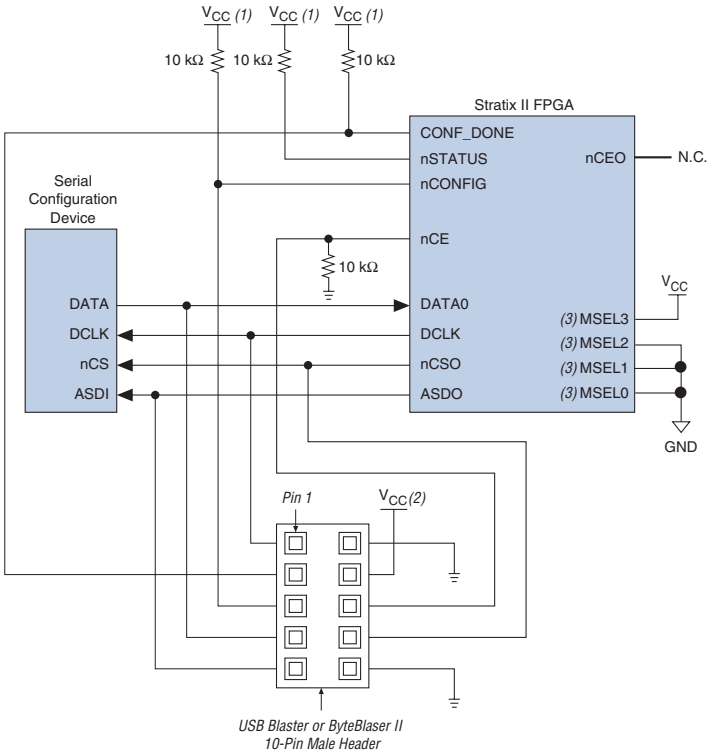
Programming Serial Configuration Devices

Serial configuration devices are non-volatile, flash-memory-based devices. You can program these devices in-system using the USB-Blaster™ or ByteBlaster™ II download cable. Alternatively, you can program them using the Altera Programming Unit (APU), supported third-party programmers, or a microprocessor with the SRunner software driver.

You can perform in-system programming of serial configuration devices via the AS programming interface. During in-system programming, the download cable disables device access to the AS interface by driving the nCE pin high. Stratix II and Stratix II GX devices are also held in reset by a low level on nCONFIG. After programming is complete, the download cable releases nCE and nCONFIG, allowing the pull-down and pull-up resistors to drive GND and V_{CC}, respectively. Figure 7–15 shows the download cable connections to the serial configuration device.

For more information on the USB Blaster download cable, refer to the *USB-Blaster Download Cable User Guide*. For more information on the ByteBlaster II cable, refer to the *ByteBlaster II Download Cable User Guide*.

Figure 7–15. In-System Programming of Serial Configuration Devices



Notes to Figure 7–15:

(1) Connect these pull-up resistors to 3.3-V supply.

(2) Power up the ByteBlaster II cable's V_{CC} with a 3.3-V supply.

(3) If using an EPCS4 device, `MSEL[3..0]` should be set to `1101`. Refer to Table 7–11 for more details.

You can program serial configuration devices with the Quartus II software with the Altera programming hardware (APU) and the appropriate configuration device programming adapter. The EPCS1 and EPCS4 devices are offered in an eight-pin small outline integrated circuit (SOIC) package.

In production environments, serial configuration devices can be programmed using multiple methods. Altera programming hardware or other third-party programming hardware can be used to program blank serial configuration devices before they are mounted onto printed circuit boards (PCBs). Alternatively, you can use an on-board microprocessor to program the serial configuration device in-system using C-based software drivers provided by Altera.

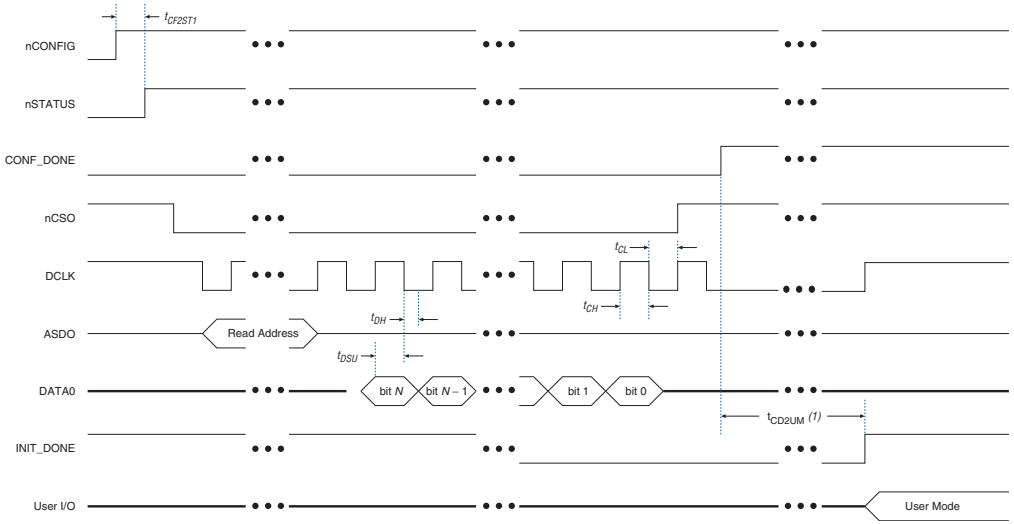
A serial configuration device can be programmed in-system by an external microprocessor using SRunner. SRunner is a software driver developed for embedded serial configuration device programming, which can be easily customized to fit in different embedded systems. SRunner is able to read a raw programming data (.**rpd**) file and write to the serial configuration devices. The serial configuration device programming time using SRunner is comparable to the programming time with the Quartus II software.

For more information about SRunner, refer to *AN 418: SRunner: An Embedded Solution for Serial Configuration Device Programming* and the source code on the Altera web site at www.altera.com.

For more information on programming serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in the *Configuration Handbook*.

Figure 7–16 shows the timing waveform for the AS configuration scheme using a serial configuration device.

Figure 7–16. AS Configuration Timing



Note to Figure 7–16:

(1) The initialization clock can come from the Stratix II or Stratix II GX device's internal oscillator or the CLKUSR pin.

Table 7–13 shows the AS timing parameters for Stratix II devices.

Table 7–13. AS Timing Parameters for Stratix II Devices

Symbol	Parameter	Condition	Minimum	Typical	Maximum
t_{CF2ST1}	nCONFIG high to nSTATUS high				100
t_{DSU}	Data setup time before falling edge on DCLK		7		
t_{DH}	Data hold time after falling edge on DCLK		0		
t_{CH}	DCLK high time		10		
t_{CL}	DCLK low time		10		
t_{CD2UM}	CONF_DONE high to user mode		20		100

Passive Serial Configuration

PS configuration of Stratix II and Stratix II GX devices can be performed using an intelligent host, such as a MAX II device or microprocessor with flash memory, an Altera configuration device, or a download cable. In the PS scheme, an external host (MAX II device, embedded processor, configuration device, or host PC) controls configuration. Configuration data is clocked into the target Stratix II or Stratix II GX device via the DATA0 pin at each rising edge of DCLK.

The Stratix II and Stratix II GX decompression and design security features are fully available when configuring your Stratix II or Stratix II GX device using PS mode.

Table 7–14 shows the MSEL pin settings when using the PS configuration scheme.

Table 7–14. Stratix II and Stratix II GX MSEL Pin Settings for PS Configuration Schemes

Configuration Scheme	MSEL3	MSEL2	MSEL1	MSEL0
PS	0	0	1	0
PS when using Remote System Upgrade *(1)*	0	1	1	0

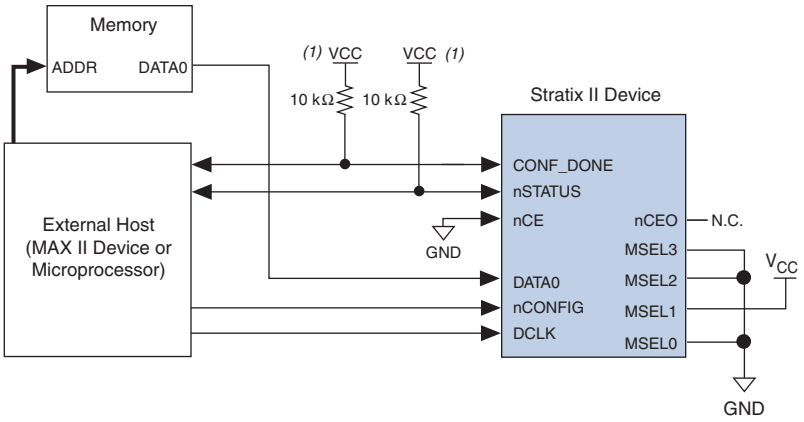
Note to Table 7–14:

(1) This scheme requires that you drive the RUnLU pin to specify either remote update or local update. For more information about remote system upgrade in Stratix II devices, refer to the *Remote System Upgrades With Stratix II & Stratix II GX Devices* chapter in volume 2 of the *Stratix II Device Handbook* or the *Remote System Upgrades With Stratix II & Stratix II GX Devices* chapter in volume 2 of the *Stratix II GX Device Handbook*.

PS Configuration Using a MAX II Device as an External Host

In the PS configuration scheme, a MAX II device can be used as an intelligent host that controls the transfer of configuration data from a storage device, such as flash memory, to the target Stratix II or Stratix II GX device. Configuration data can be stored in RBF, HEX, or TTF format. Figure 7–17 shows the configuration interface connections between a Stratix II or Stratix II GX device and a MAX II device for single device configuration.

Figure 7–17. Single Device PS Configuration Using an External Host



Note to Figure 7–17:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for the device. V_{CC} should be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

Upon power-up, Stratix II and Stratix II GX devices go through a POR. The POR delay is dependent on the PORSEL pin setting; when PORSEL is driven low, the POR time is approximately 100 ms, if PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device resets, holds nSTATUS low, and tri-states all user I/O pins. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

You can hold nConfig low in order to stop device configuration.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *Stratix II Device Handbook* or the *Stratix II GX Device Handbook*.

The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration, the MAX II device must generate a low-to-high transition on the nCONFIG pin.

V_{CCINT}, V_{CCIO}, and V_{CCPD} of the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. Once nSTATUS is released, the device is ready to receive configuration data and the configuration stage begins. When nSTATUS is pulled high, the MAX II device should place the configuration data one bit at a time on the DATA0 pin. If you are using configuration data in RBF, HEX, or TTF format, you must send the least significant bit (LSB) of each data byte first. For example, if the RBF contains the byte sequence `02 1B EE 01 FA`, the serial bitstream you should transmit to the device is `0100-0000 1101-1000 0111-0111 1000-0000 0101-1111`.

The Stratix II and Stratix II GX devices receive configuration data on the DATA0 pin and the clock is received on the DCLK pin. Data is latched into the device on the rising edge of DCLK. Data is continuously clocked into the target device until CONF_DONE goes high. After the device has received all configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by an external 10-kΩ pull-up resistor. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

In Stratix II and Stratix II GX devices, the initialization clock source is either the internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Stratix II or Stratix II GX device provides itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device. Driving DCLK to the device after configuration is complete does not affect device operation.

You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. The **Enable user-supplied start-up clock** (**CLKUSR**) option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR will not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, CLKUSR will be enabled after the time specified as t_{CD2CU}. After this time period elapses, Stratix II and Stratix II GX devices require 299 clock cycles to initialize properly and enter user mode. Stratix II and Stratix II GX devices support a CLKUSR f_{MAX} of 100 MHz.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used it will be high due to an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low. When initialization is complete, the INIT_DONE pin will be released and pulled high. The MAX II device must be able to detect this low-to-high transition which signals the device has entered user mode. When initialization is complete, the device enters user mode. In user-mode, the user I/O pins will no longer have weak pull-up resistors and will function as assigned in your design.

To ensure DCLK and DATA0 are not left floating at the end of configuration, the MAX II device must drive them either high or low, whichever is convenient on your board. The DATA[0] pin is available as a user I/O pin after configuration. When the PS scheme is chosen in the Quartus II software, as a default this I/O pin is tri-stated in user mode and should be driven by the MAX II device. To change this default option in the Quartus II software, select the **Dual-Purpose Pins** tab of the **Device & Pin Options** dialog box.

The configuration clock (DCLK) speed must be below the specified frequency to ensure correct configuration. No maximum DCLK period exists, which means you can pause configuration by halting DCLK for an indefinite amount of time.

If an error occurs during configuration, the device drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the MAX II device that there is an error. If the **Auto-restart configuration after error** option (available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box) is turned on, the Stratix II or Stratix II GX device releases nSTATUS after a reset time-out period (maximum of 100 µs). After nSTATUS is released and

pulled high by a pull-up resistor, the MAX II device can try to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

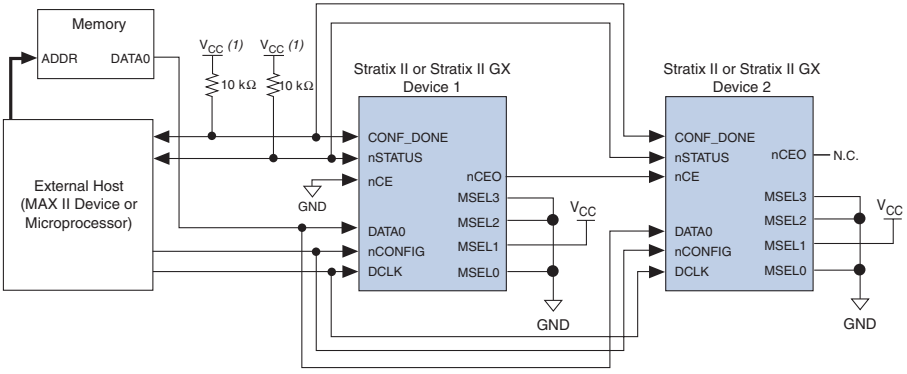
The MAX II device can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The CONF_DONE pin must be monitored by the MAX II device to detect errors and determine when programming completes. If all configuration data is sent, but CONF_DONE or INIT_DONE have not gone high, the MAX II device must reconfigure the target device.

If the optional CLKUSR pin is being used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure that CLKUSR continues toggling during the time nSTATUS is low (maximum of 100 µs).

When the device is in user-mode, you can initiate a reconfiguration by transitioning the nCONFIG pin low-to-high. The nCONFIG pin must be low for at least 2 µs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Once nCONFIG returns to a logic high level and nSTATUS is released by the device, reconfiguration begins.

Figure 7–18 shows how to configure multiple devices using a MAX II device. This circuit is similar to the PS configuration circuit for a single device, except Stratix II or Stratix II GX devices are cascaded for multi-device configuration.

Figure 7–18. Multi-Device PS Configuration Using an External Host



Note to Figure 7–18:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain. V_{CC} should be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

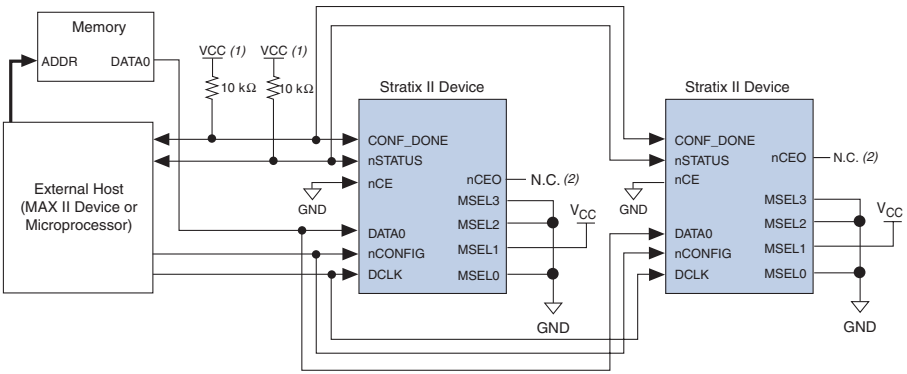
In multi-device PS configuration the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration within one clock cycle. Therefore, the transfer of data destinations is transparent to the MAX II device. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Since all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

If the **Auto-restart configuration after error** option is turned on, the devices release their nSTATUS pins after a reset time-out period (maximum of 100 µs). After all nSTATUS pins are released and pulled high, the MAX II device can try to reconfigure the chain without needing to pulse nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

In your system, you can have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete configuration at the same time. Figure 7–19 shows multi-device PS configuration when both Stratix II or Stratix II GX devices are receiving the same configuration data.

Figure 7–19. Multiple-Device PS Configuration When Both devices Receive the Same Data



Notes to Figure 7–19:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain. V_{CC} should be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

(2) The nCEO pins of both devices are left unconnected when configuring the same configuration data into multiple devices.

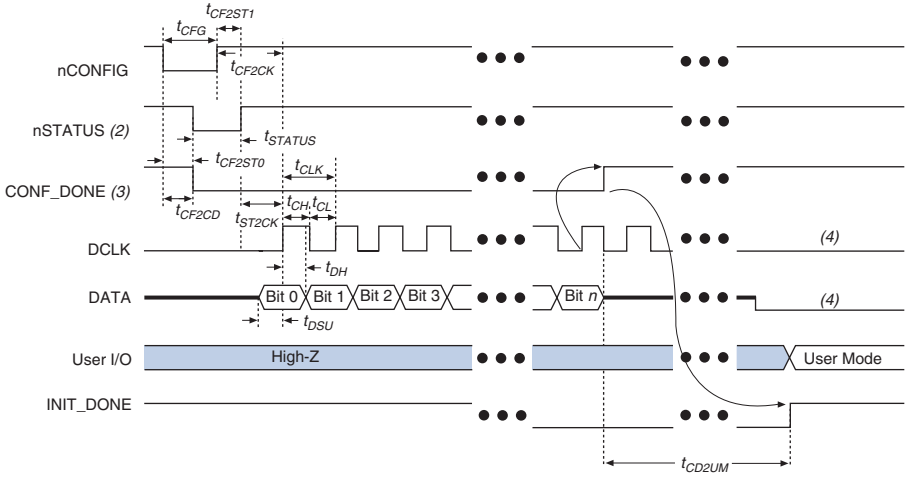
You can use a single configuration chain to configure Stratix II or Stratix II GX devices with other Altera devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information on configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook*.

PS Configuration Timing

Figure 7–20 shows the timing waveform for PS configuration when using a MAX II device as an external host.

Figure 7–20. PS Configuration Timing Waveform *Note (1)*



Notes to Figure 7–20:

(1) The beginning of this waveform shows the device in user-mode. In user-mode, nCONFIG, nSTATUS and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.
(2) Upon power-up, the Stratix II or Stratix II GX device holds nSTATUS low for the time of the POR delay.
(3) Upon power-up, before and during configuration, CONF_DONE is low.
(4) DCLK should not be left floating after configuration. It should be driven high or low, whichever is more convenient. DATA[0] is available as a user I/O pin after configuration and the state of this pin depends on the dual-purpose pin settings.

Table 7–15 defines the timing parameters for Stratix II and Stratix II GX devices for PS configuration.

Table 7–15. PS Timing Parameters for Stratix II and Stratix II GX Devices (Part 1 of 2)

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		800	ns

Table 7–15. PS Timing Parameters for Stratix II and Stratix II GX Devices (Part 2 of 2)

Symbol	Parameter	Min	Max	Units
t_{CFG}	nCONFIG low pulse width	2		µs
t_{STATUS}	nSTATUS low pulse width	10	100 *(1)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high		100 *(1)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	100		µs
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2		µs
t_{DSU}	Data setup time before rising edge on DCLK	5		ns
t_{DH}	Data hold time after rising edge on DCLK	0		ns
t_{CH}	DCLK high time	4		ns
t_{CL}	DCLK low time	4		ns
t_{CLK}	DCLK period	10		ns
f_{MAX}	DCLK frequency		100	MHz
t_R	Input rise time		40	ns
t_F	Input fall time		40	ns
t_{CD2UM}	CONF_DONE high to user mode *(2)*	20	100	µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period		
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	t_{CD2CU} + (299 × CLKUSR period)		

Notes to Table 7–15:

(1) This value is applicable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(2) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting the device.

Device configuration options and how to create configuration files are discussed further in *Software Settings* in volume 2 of the *Configuration Handbook*.

An example PS design that uses a MAX II device as the external host for configuration will be available when devices are available.

PS Configuration Using a Microprocessor

In the PS configuration scheme, a microprocessor can control the transfer of configuration data from a storage device, such as flash memory, to the target Stratix II or Stratix II GX device.

All information in the "PS Configuration Using a MAX II Device as an External Host" section is also applicable when using a microprocessor as an external host. Refer to that section for all configuration and timing information.

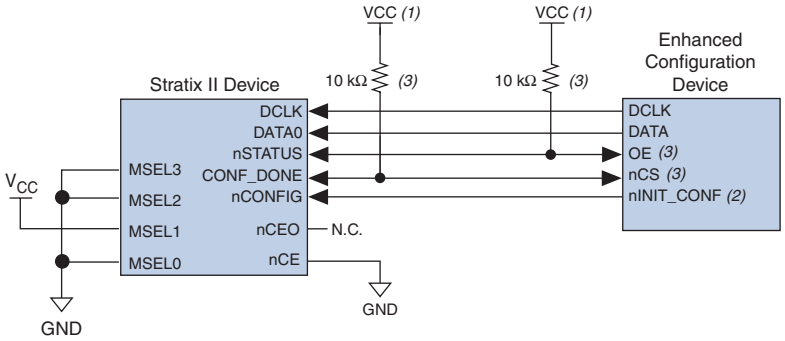
PS Configuration Using a Configuration Device

You can use an Altera configuration device, such as an enhanced configuration device, to configure Stratix II and Stratix II GX devices using a serial configuration bitstream. Configuration data is stored in the configuration device. Figure 7–21 shows the configuration interface connections between a Stratix II or Stratix II GX device and a configuration device.

The figures in this chapter only show the configuration-related pins and the configuration pin connections between the configuration device and the device.

For more information on the enhanced configuration device and flash interface pins (such as `PGM[2..0]`, `EXCLK`, `PORSEL`, `A[20..0]`, and `DQ[15..0]`), refer to the *Enhanced Configuration Devices (EPC4, EPC8, EPC16, EPCS64, and EPCS128) Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

Figure 7–21. Single Device PS Configuration Using an Enhanced Configuration Device



Notes to Figure 7–21:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

The value of the internal pull-up resistors on the enhanced configuration devices can be found in the Operating Conditions table of the *Enhanced Configuration Devices (EPC4, EPC8, & EPC16) Data Sheet* chapter in volume 2 of the *Configuration Handbook* or the *Configuration Devices for SRAM-based LUT Devices Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

When using enhanced configuration devices, nCONFIG of the device can be connected to nINIT_CONF of the configuration device, which allows the INIT_CONF JTAG instruction to initiate device configuration. The nINIT_CONF pin does not need to be connected if its functionality is not used. An internal pull-up resistor on the nINIT_CONF pin is always active in enhanced configuration devices, which means an external pull-up resistor should not be used if nCONFIG is tied to nINIT_CONF.

Upon power-up, the Stratix II and Stratix II GX devices go through a POR. The POR delay is dependent on the PORSEL pin setting. When PORSEL is driven low, the POR time is approximately 100 ms. If PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device will reset, hold nSTATUS low, and tri-state all user I/O pins. The configuration device also goes through a POR delay to allow the power supply to stabilize. The POR time for EPC2 devices is 200 ms (maximum). The POR time for enhanced configuration devices can be set to either

100 ms or 2 ms, depending on its PORSEL pin setting. If the PORSEL pin is connected to GND, the POR delay is 100 ms. If the PORSEL pin is connected to V_{CC}, the POR delay is 2 ms. During this time, the configuration device drives its OE pin low. This low signal delays configuration because the OE pin is connected to the target device's nSTATUS pin.

When selecting a POR time, you need to ensure that the device completes power-up before the enhanced configuration device exits POR. Altera recommends that you choose a POR time for the Stratix II or Stratix II GX device of 12 ms, while selecting a POR time for the enhanced configuration device of 100 ms.

When both devices complete POR, they release their open-drain OE or nSTATUS pin, which is then pulled high by a pull-up resistor. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *DC & Switching Characteristics* chapter in volume 2 of the *Stratix II Device Handbook* or the *DC & Switching Characteristics* chapter in volume 2 of the *Stratix II GX Device Handbook*.

When the power supplies have reached the appropriate operating voltages, the target device senses the low-to-high transition on nCONFIG and initiates the configuration cycle. The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. The beginning of configuration can be delayed by holding the nCONFIG or nSTATUS pin low.

To begin configuration, power the V_{CCINT}, V_{CCIO}, and V_{CCPD} voltages (for the banks where the configuration and JTAG pins reside) to the appropriate voltage levels.

When nCONFIG goes high, the device comes out of reset and releases the nSTATUS pin, which is pulled high by a pull-up resistor. Enhanced configuration and EPC2 devices have an optional internal pull-up resistor on the OE pin. This option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If this internal pull-up resistor is not used, an external 10-kΩ pull-up resistor on the OE-nSTATUS line is required. Once nSTATUS is released, the device is ready to receive configuration data and the configuration stage begins.

When nSTATUS is pulled high, OE of the configuration device also goes high and the configuration device clocks data out serially to the device using the Stratix II or Stratix II GX device's internal oscillator. The Stratix II and Stratix II GX devices receive configuration data on the DATA0 pin and the clock is received on the DCLK pin. Data is latched into the device on the rising edge of DCLK.

After the device has received all configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by a pull-up resistor. Since CONF_DONE is tied to the configuration device's nCS pin, the configuration device is disabled when CONF_DONE goes high. Enhanced configuration and EPC2 devices have an optional internal pull-up resistor on the nCS pin. This option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If this internal pull-up resistor is not used, an external 10-kΩ pull-up resistor on the nCS-CONF_DONE line is required. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin.

In Stratix II and Stratix II GX devices, the initialization clock source is either the internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If you are using internal oscillator, the Stratix II or Stratix II GX device supplies itself with enough clock cycles for proper initialization. You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. You can turn on the **Enable user-supplied start-up clock** (**CLKUSR**) option in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR will not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, CLKUSR will be enabled after the time specified as t_{CD2CU}. After this time period elapses, the Stratix II and Stratix II GX devices require 299 clock cycles to initialize properly and enter user mode. Stratix II and Stratix II GX devices support a CLKUSR f_{MAX} of 100 MHz.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If you are using the INIT_DONE pin, it will be high due to an external 10-kΩpull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. This low-to-high transition signals that the device has entered user mode. In user-mode, the user I/O

pins will no longer have weak pull-up resistors and will function as assigned in your design. Enhanced configuration devices and EPC2 devices drive DCLK low and DATA0 high at the end of configuration.

If an error occurs during configuration, the device drives its nSTATUS pin low, resetting itself internally. Since the nSTATUS pin is tied to OE, the configuration device will also be reset. If the **Auto-restart configuration after error** option, available in the Quartus II software, from the **General** tab of the **Device & Pin Options** dialog box is turned on, the device automatically initiates reconfiguration if an error occurs. The Stratix II and Stratix II GX devices release the nSTATUS pin after a reset time-out period (maximum of 100 µs). When the nSTATUS pin is released and pulled high by a pull-up resistor, the configuration device reconfigures the chain. If this option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 2 µs to restart configuration. The external system can pulse nCONFIG if nCONFIG is under system control rather than tied to V_{CC}.

In addition, if the configuration device sends all of its data and then detects that CONF_DONE has not gone high, it recognizes that the device has not configured successfully. Enhanced configuration devices wait for 64 DCLK cycles after the last configuration bit was sent for CONF_DONE to reach a high state. EPC2 devices wait for 16 DCLK cycles. In this case, the configuration device pulls its OE pin low, driving the target device's nSTATUS pin low. If the **Auto-restart configuration after error** option is set in the software, the target device resets and then releases its nSTATUS pin after a reset time-out period (maximum of 100 µs). When nSTATUS returns to a logic high level, the configuration device tries to reconfigure the device.

When CONF_DONE is sensed low after configuration, the configuration device recognizes that the target device has not configured successfully. Therefore, your system should not pull CONF_DONE low to delay initialization. Instead, use the CLKUSR option to synchronize the initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain will initialize together if their CONF_DONE pins are tied together.

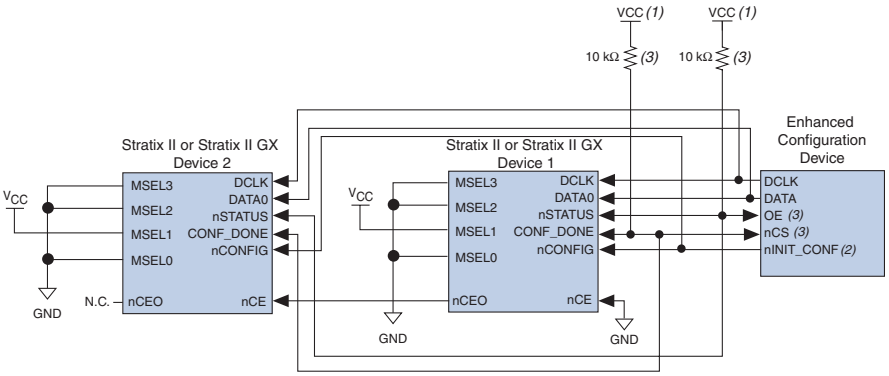
If you are using the optional CLKUSR pin and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure that CLKUSR continues toggling during the time nSTATUS is low (maximum of 100 µs).

When the device is in user-mode, pulling the nCONFIG pin low initiates a reconfiguration. The nCONFIG pin should be low for at least 2 µs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Because CONF_DONE is pulled low, this

activates the configuration device because it sees its nCS pin drive low. Once nCONFIG returns to a logic high level and nSTATUS is released by the device, reconfiguration begins.

Figure 7–22 shows how to configure multiple devices with an enhanced configuration device. This circuit is similar to the configuration device circuit for a single device, except Stratix II or Stratix II GX devices are cascaded for multi-device configuration.

Figure 7–22. Multi-Device PS Configuration Using an Enhanced Configuration Device



Notes to Figure 7–22:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

Enhanced configuration devices cannot be cascaded.

When performing multi-device configuration, you must generate the configuration device's POF from each project's SOF. You can combine multiple SOFs using the **Convert Programming Files** window in the Quartus II software.

For more information on how to create configuration files for multi-device configuration chains, refer to the *Software Settings* chapter of the *Configuration Handbook*.

In multi-device PS configuration, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, prompting the second device to begin configuration. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device.

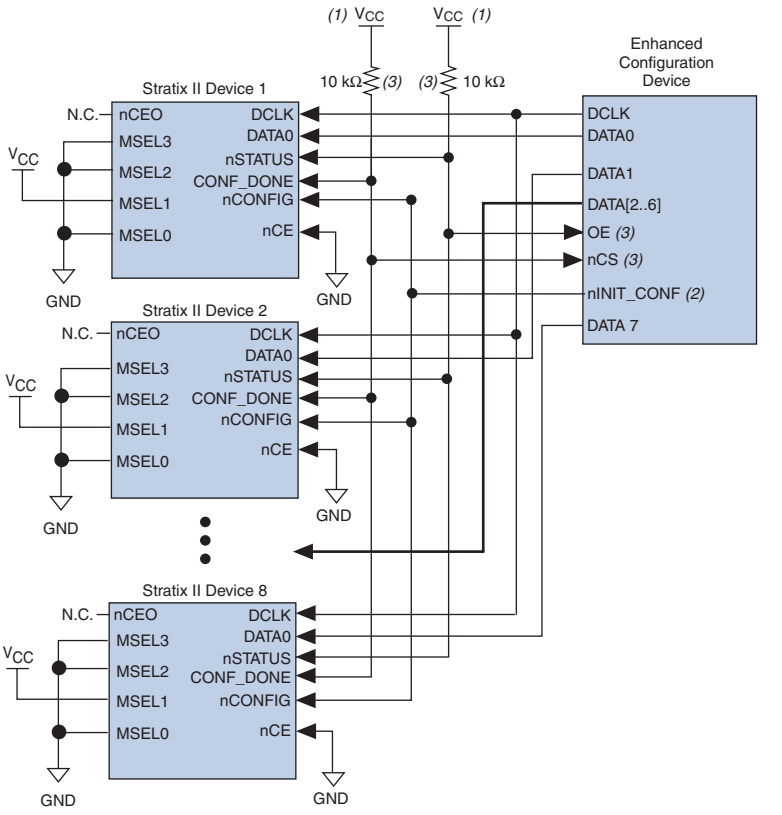
When configuring multiple devices, configuration does not begin until all devices release their OE or nSTATUS pins. Similarly, since all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Since all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This low signal drives the OE pin low on the enhanced configuration device and drives nSTATUS low on all devices, causing them to enter a reset state. This behavior is similar to a single device detecting an error.

If the **Auto-restart configuration after error** option is turned on, the devices will automatically initiate reconfiguration if an error occurs. The devices will release their nSTATUS pins after a reset time-out period (maximum of 100 µs). When all the nSTATUS pins are released and pulled high, the configuration device tries to reconfigure the chain. If the **Auto-restart configuration after error** option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 2 µs to restart configuration. The external system can pulse nCONFIG if nCONFIG is under system control rather than tied to V_{CC}.

The enhanced configuration devices also support parallel configuration of up to eight devices. The n-bit (n = 1, 2, 4, or 8) PS configuration mode allows enhanced configuration devices to concurrently configure devices or a chain of devices. In addition, these devices do not have to be the same device family or density as they can be any combination of Altera devices. An individual enhanced configuration device DATA line is available for each targeted device. Each DATA line can also feed a daisy chain of devices. Figure 7–23 shows how to concurrently configure multiple devices using an enhanced configuration device.

Figure 7–23. Concurrent PS Configuration of Multiple Devices Using an Enhanced Configuration Device



Notes to Figure 7–23:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

The Quartus II software only allows the selection of n-bit PS configuration modes, where n must be 1, 2, 4, or 8. However, you can use these modes to configure any number of devices from 1 to 8. When configuring SRAM-based devices using n-bit PS modes, use Table 7–16 to select the appropriate configuration mode for the fastest configuration times.

Table 7–16. Recommended Configuration Using n-Bit PS Modes

Number of Devices *(1)*	Recommended Configuration Mode
1	1-bit PS
2	2-bit PS
3	4-bit PS
4	4-bit PS
5	8-bit PS
6	8-bit PS
7	8-bit PS
8	8-bit PS

Note to Table 7–16:

(1) Assume that each DATA line is only configuring one device, not a daisy chain of devices.

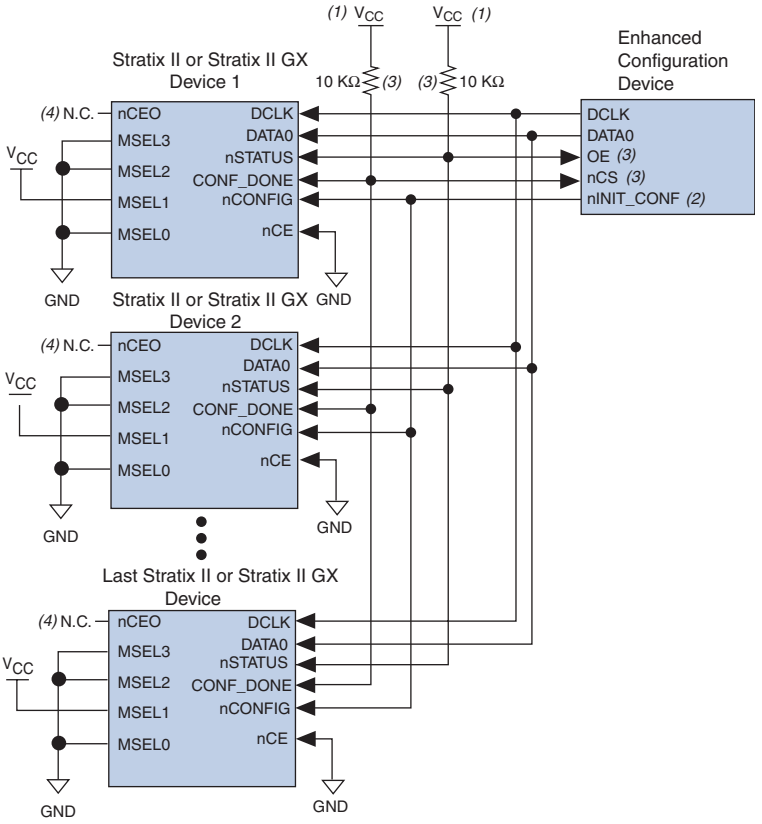
For example, if you configure three devices, you would use the 4-bit PS mode. For the DATA0, DATA1, and DATA2 lines, the corresponding SOF data is transmitted from the configuration device to the device. For DATA3, you can leave the corresponding Bit3 line blank in the Quartus II software. On the PCB, leave the DATA3 line from the enhanced configuration device unconnected.

Alternatively, you can daisy chain two devices to one DATA line while the other DATA lines drive one device each. For example, you could use the 2-bit PS mode to drive two devices with DATA Bit0 (two EP2S15 devices) and the third device (EP2S30 device) with DATA Bit1. This 2-bit PS configuration scheme requires less space in the configuration flash memory, but can increase the total system configuration time.

A system may have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete

configuration at the same time. Figure 7–24 shows multi-device PS configuration when the Stratix II or Stratix II GX devices are receiving the same configuration data.

Figure 7–24. Multiple-Device PS Configuration Using an Enhanced Configuration Device When devices Receive the Same Data



Notes to Figure 7–24:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

(4) The nCEO pins of all devices are left unconnected when configuring the same configuration data into multiple devices.

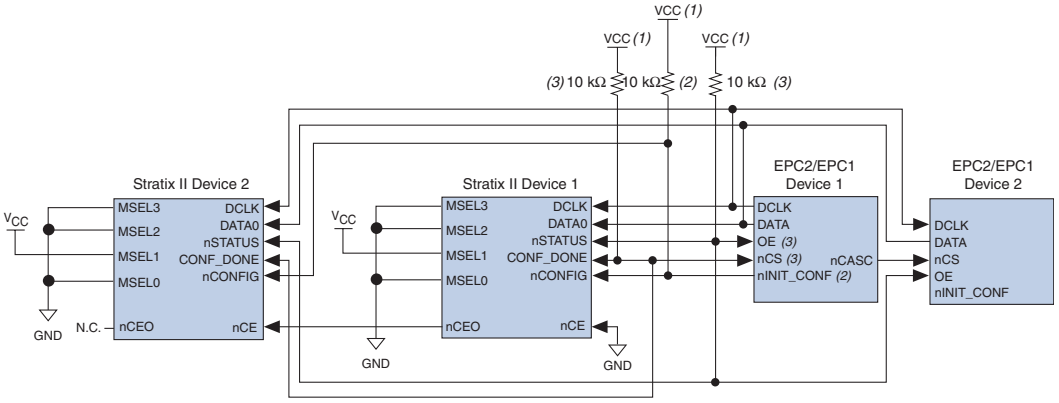
You can cascade several EPC2 devices to configure multiple Stratix II or Stratix II GX devices. The first configuration device in the chain is the master configuration device, while the subsequent devices are the slave devices. The master configuration device sends DCLK to the Stratix II or Stratix II GX devices and to the slave configuration devices. The first EPC device's nCS pin is connected to the CONF_DONE pins of the devices, while its nCASC pin is connected to nCS of the next configuration device in the chain. The last device's nCS input comes from the previous device, while its nCASC pin is left floating. When all data from the first configuration device is sent, it drives nCASC low, which in turn drives nCS on the next configuration device. A configuration device requires less than one clock cycle to activate a subsequent configuration device, so the data stream is uninterrupted.

Enhanced configuration devices cannot be cascaded.

Because all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, the master configuration device stops configuration for the entire chain and the entire chain must be reconfigured. For example, if the master configuration device does not detect CONF_DONE going high at the end of configuration, it resets the entire chain by pulling its OE pin low. This low signal drives the OE pin low on the slave configuration device(s) and drives nSTATUS low on all devices, causing them to enter a reset state. This behavior is similar to the device detecting an error in the configuration data.

Figure 7–25 shows how to configure multiple devices using cascaded EPC2 devices.

Figure 7–25. Multi-Device PS Configuration Using Cascaded EPC2 Devices



Notes to Figure 7–25:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used.

(3) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. External 10-kΩ pull-up resistors should be used. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

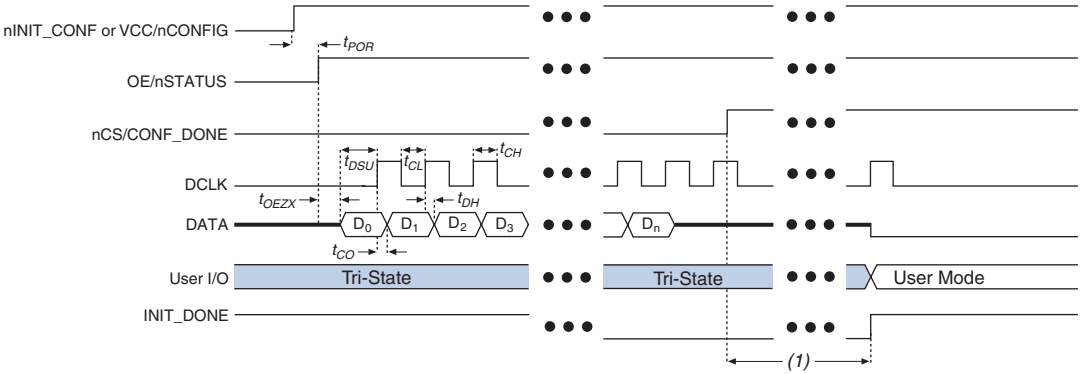
When using enhanced configuration devices or EPC2 devices, nCONFIG of the device can be connected to nINIT_CONF of the configuration device, allowing the INIT_CONF JTAG instruction to initiate device configuration. The nINIT_CONF pin does not need to be connected if its functionality is not used. An internal pull-up resistor on the nINIT_CONF pin is always active in the enhanced configuration devices and the EPC2 devices, which means that you shouldn't be using an external pull-up resistor if nCONFIG is tied to nINIT_CONF. If you are using multiple EPC2 devices to configure a Stratix II or Stratix II GX device(s), only the first EPC2 has its nINIT_CONF pin tied to the device's nCONFIG pin.

You can use a single configuration chain to configure Stratix II or Stratix II GX devices with other Altera devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information on configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in the *Configuration Handbook*.

Figure 7–26 shows the timing waveform for the PS configuration scheme using a configuration device.

Figure 7–26. Stratix II and Stratix II GX PS Configuration Using a Configuration Device Timing Waveform



Note to Figure 7–26:

(1) The initialization clock can come from the Stratix II or Stratix II GX device's internal oscillator or the CLKUSR pin.

For timing information, refer to the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* chapter in volume 2 of the *Configuration Handbook* or the *Configuration Devices for SRAM-Based LUT Devices Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

Device configuration options and how to create configuration files are discussed further in the *Software Settings* chapter in volume 2 of the *Configuration Handbook*.

PS Configuration Using a Download Cable

In this section, the generic term "download cable" includes the Altera USB-Blaster™ universal serial bus (USB) port download cable, MasterBlaster™ serial/USB communications cable, ByteBlaster™ II parallel port download cable, and the ByteBlaster MV parallel port download cable.

In PS configuration with a download cable, an intelligent host (such as a PC) transfers data from a storage device to the device via the USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV cable.

Upon power-up, the Stratix II and Stratix II GX devices go through a POR. The POR delay is dependent on the PORSEL pin setting. When PORSEL is driven low, the POR time is approximately 100 ms. If PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device will reset, hold nSTATUS low, and tri-state all user I/O pins. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *DC & Switching Characteristics* chapter in volume 1 of the *Stratix II Device Handbook* and the *DC & Switching Characteristics* chapter in volume 1 of the *Stratix II GX Device Handbook.*

The configuration cycle consists of three stages: reset, configuration and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration in this scheme, the download cable generates a low-to-high transition on the nCONFIG pin.

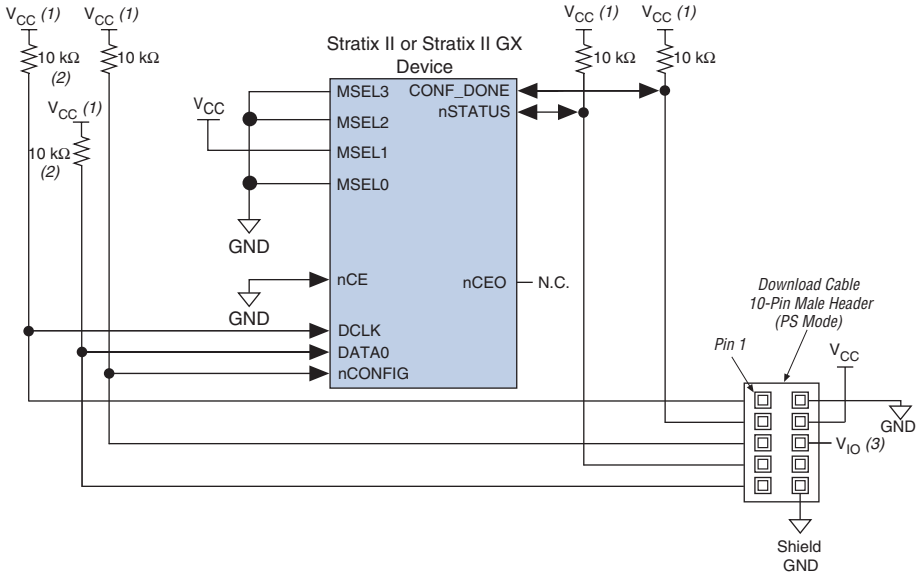
To begin configuration, power the V_{CCINT}, V_{CCIO}, and V_{CCPD} voltages (for the banks where the configuration and JTAG pins reside) to the appropriate voltage levels.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. Once nSTATUS is released the device is ready to receive configuration data and the configuration stage begins. The programming hardware or download cable then places the configuration data one bit at a time on the device's DATA0 pin. The configuration data is clocked into the target device until CONF_DONE goes high. The CONF_DONE pin must have an external 10-kΩpull-up resistor in order for the device to initialize.

When using a download cable, setting the **Auto-restart configuration after error** option does not affect the configuration cycle because you must manually restart configuration in the Quartus II software when an error occurs. Additionally, the **Enable user-supplied start-up clock** (**CLKUSR**) option has no affect on the device initialization since this option is disabled in the SOF when programming the device using the Quartus II programmer and download cable. Therefore, if you turn on the CLKUSR option, you do not need to provide a clock on CLKUSR when you are configuring the device with the Quartus II programmer and a

download cable. Figure 7–27 shows PS configuration for Stratix II or Stratix II GX devices using a USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV cable.

Figure 7–27. PS Configuration Using a USB Blaster, MasterBlaster, ByteBlaster II or ByteBlasterMV Cable



Notes to Figure 7–27:

(1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster (V_{IO} pin), ByteBlaster II or ByteBlasterMV cable.

(2) The pull-up resistors on DATA0 and DCLK are only needed if the download cable is the only configuration scheme used on your board. This ensures that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, the pull-up resistors on DATA0 and DCLK are not needed.

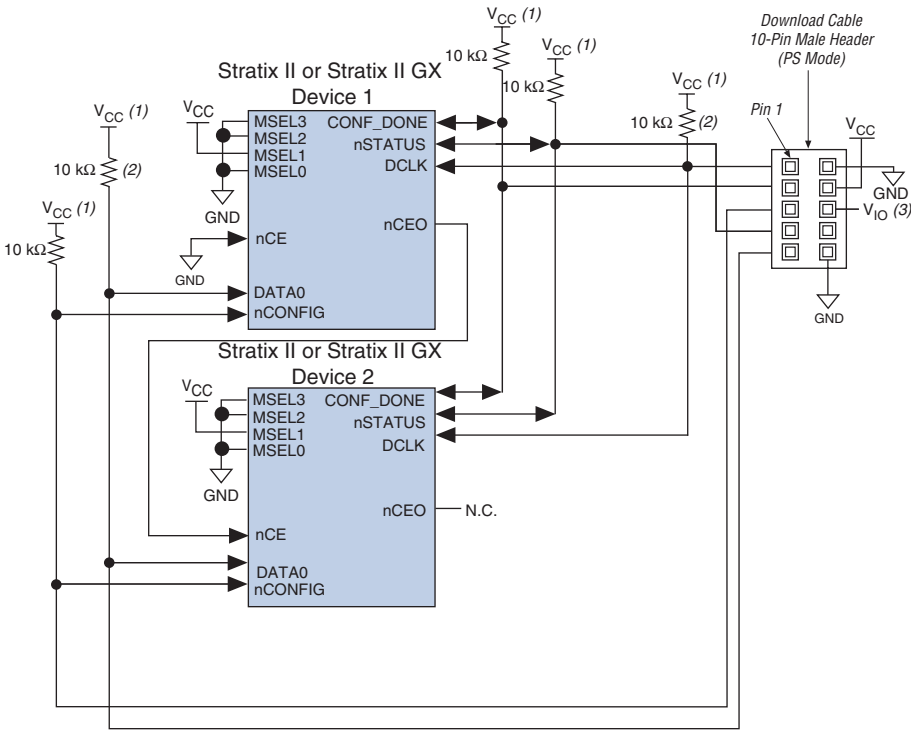
(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. Refer to the *MasterBlaster Serial/USB Communications Cable User Guide* for this value. In the ByteBlasterMV cable, this pin is a no connect. In the USB Blaster and ByteBlaster II cables, this pin is connected to nCE when it is used for active serial programming, otherwise it is a no connect.

You can use a download cable to configure multiple Stratix II or Stratix II GX devices by connecting each device's nCEO pin to the subsequent device's nCE pin. The first device's nCE pin is connected to GND while its nCEO pin is connected to the nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. All other configuration pins, nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE are connected to every device in the chain. Because all CONF_DONE pins are tied together, all devices in the chain initialize and enter user mode at the same time.

In addition, because the nSTATUS pins are tied together, the entire chain halts configuration if any device detects an error. The **Auto-restart configuration after error** option does not affect the configuration cycle because you must manually restart configuration in the Quartus II software when an error occurs.

Figure 7–28 shows how to configure multiple Stratix II or Stratix II GX devices with a download cable.

Figure 7–28. Multi-Device PS Configuration using a USB Blaster, MasterBlaster, ByteBlaster II or ByteBlasterMV Cable



Notes to Figure 7–28:

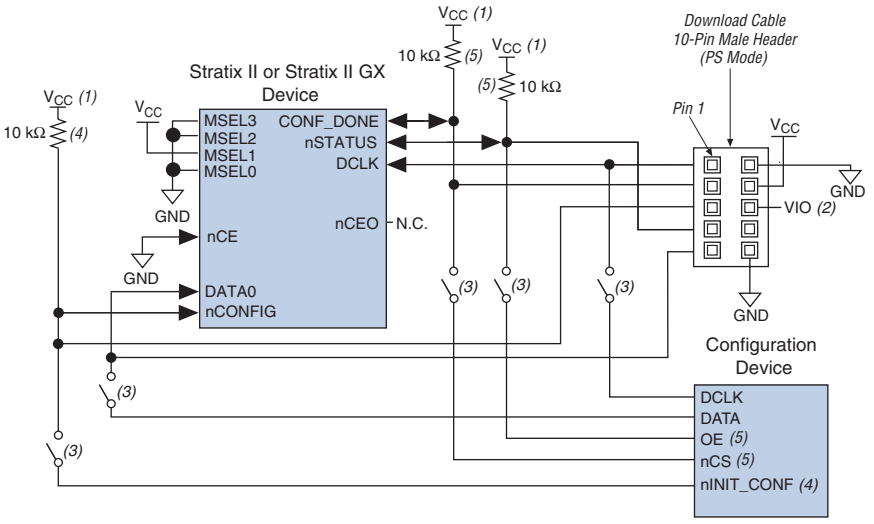
(1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster (V_{IO} pin), ByteBlaster II or ByteBlasterMV cable.

(2) The pull-up resistors on DATA0 and DCLK are only needed if the download cable is the only configuration scheme used on your board. This is to ensure that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, the pull-up resistors on DATA0 and DCLK are not needed.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. Refer to the *MasterBlaster Serial/USB Communications Cable User Guide* for this value. In the ByteBlasterMV cable, this pin is a no connect. In the USB Blaster and ByteBlaster II cables, this pin is connected to nCE when it is used for active serial programming, otherwise it is a no connect.

If you are using a download cable to configure device(s) on a board that also has configuration devices, electrically isolate the configuration device from the target device(s) and cable. One way of isolating the configuration device is to add logic, such as a multiplexer, that can select between the configuration device and the cable. The multiplexer chip allows bidirectional transfers on the nSTATUS and CONF_DONE signals. Another option is to add switches to the five common signals (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) between the cable and the configuration device. The last option is to remove the configuration device from the board when configuring the device with the cable. Figure 7–29 shows a combination of a configuration device and a download cable to configure an device.

Figure 7–29. PS Configuration with a Download Cable and Configuration Device Circuit



Notes to Figure 7–29:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. Refer to the *MasterBlaster Serial/USB Communications Cable User Guide* for this value. In the ByteBlasterMV cable, this pin is a no connect. In the USB Blaster and ByteBlaster II cables, this pin is connected to nCE when it is used for active serial programming, otherwise it is a no connect.

(3) You should not attempt configuration with a download cable while a configuration device is connected to a Stratix II or Stratix II GX device. Instead, you should either remove the configuration device from its socket when using the download cable or place a switch on the five common signals between the download cable and the configuration device.

(4) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) has an internal pull-up resistor that is always active. This means an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used.

(5) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-up resistors on configuration device** option when generating programming files.

For more information on how to use the USB Blaster, MasterBlaster, ByteBlaster II or ByteBlasterMV cables, refer to the following data sheets:

- *USB Blaster Download Cable User Guide*
- *MasterBlaster Serial/USB Communications Cable User Guide*
- *ByteBlaster II Download Cable User Guide*
- *ByteBlasterMV Download Cable User Guide*

Passive Parallel Asynchronous Configuration

Passive parallel asynchronous (PPA) configuration uses an intelligent host, such as a microprocessor, to transfer configuration data from a storage device, such as flash memory, to the target Stratix II or Stratix II GX device.

Configuration data can be stored in RBF, HEX, or TTF format. The host system outputs byte-wide data and the accompanying strobe signals to the device. When using PPA, pull the DCLK pin high through a 10-kΩpull-up resistor to prevent unused configuration input pins from floating.

You cannot use the Stratix II or Stratix II GX decompression and design security features if you are configuring your Stratix II or Stratix II GX device using PPA mode.

Table 7–17 shows the MSEL pin settings when using the PS configuration scheme.

Table 7–17. Stratix II and Stratix II GX MSEL Pin Settings for PPA Configuration Schemes

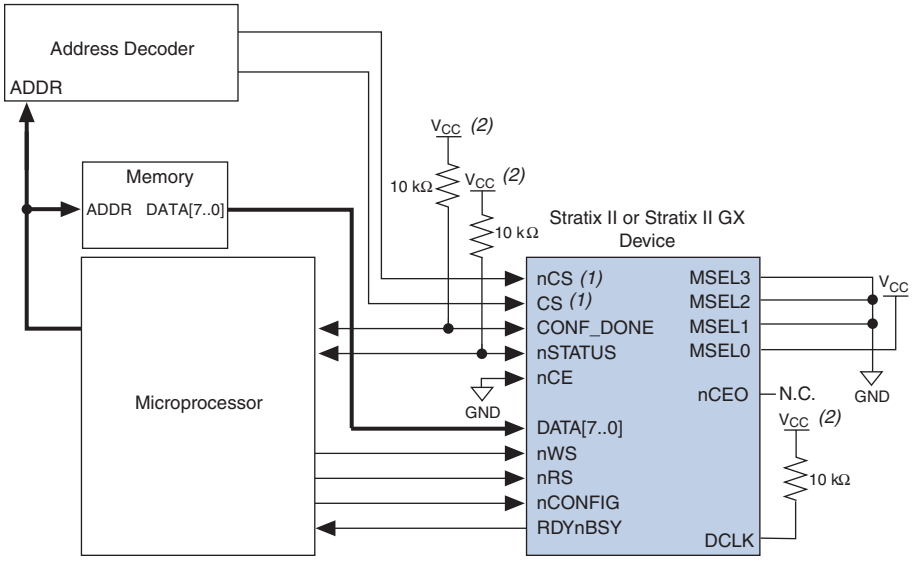
Configuration Scheme	MSEL3	MSEL2	MSEL1	MSEL0
PPA	0	0	0	1
Remote System Upgrade PPA *(1)*	0	1	0	1

Note to Table 7–17:

(1) This scheme requires that you drive the RUnLU pin to specify either remote update or local update. For more information about remote system upgrades in Stratix II and Stratix II GX devices, refer to the *Remote System Upgrades With Stratix II & Stratix II GX Devices* chapter in volume 2 of the *Stratix II Device Handbook* or the *Remote System Upgrades With Stratix II & Stratix II GX Devices* chapter in volume 2 of the *Stratix II GX Device Handbook*.

Figure 7–30 shows the configuration interface connections between the device and a microprocessor for single device PPA configuration. The microprocessor or an optional address decoder can control the device's chip select pins, nCS and CS. The address decoder allows the microprocessor to select the Stratix II or Stratix II GX device by accessing a particular address, which simplifies the configuration process. Hold the nCS and CS pins active during configuration and initialization.

Figure 7–30. Single Device PPA Configuration Using a Microprocessor



Notes to Figure 7–30:

(1) If not used, the CS pin can be connected to V_{CC} directly. If not used, the nCS pin can be connected to GND directly.

(2) The pull-up resistor should be connected to a supply that provides an acceptable input signal for the device. V_{CC} should be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

During PPA configuration, it is only required to use either the nCS or CS pin. Therefore, if you are using only one chip-select input, the other must be tied to the active state. For example, nCS can be tied to ground while CS is toggled to control configuration. The device's nCS or CS pins can be toggled during PPA configuration if the design meets the specifications set for t_{CSSU}, t_{WSP}, and t_{CSH} listed in Table 7–18.

Upon power-up, the Stratix II and Stratix II GX devices go through a POR. The POR delay is dependent on the PORSEL pin setting. When PORSEL is driven low, the POR time is approximately 100 ms. If PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device will reset, hold nSTATUS low, and tri-state all user I/O pins. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

You can hold nConfig low in order to stop device configuration.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *DC & Switching Characteristics* chapter in volume 1 of the *Stratix II Device Handbook* and the *DC & Switching Characteristics* chapter in volume 1 of the *Stratix II GX Device Handbook*.

The configuration cycle consists of three stages: reset, configuration and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration, the microprocessor must generate a low-to-high transition on the nCONFIG pin.

To begin configuration, power the V_{CCINT}, V_{CCIO}, and V_{CCPD} voltages (for the banks where the configuration and JTAG pins reside) to the appropriate voltage levels.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. Once nSTATUS is released the device is ready to receive configuration data and the configuration stage begins. When nSTATUS is pulled high, the microprocessor should then assert the target device's nCS pin low and/or CS pin high. Next, the microprocessor places an 8-bit configuration word (one byte) on the target device's DATA[7..0] pins and pulses the nWS pin low.

On the rising edge of nWS, the target device latches in a byte of configuration data and drives its RDYnBSY signal low, which indicates it is processing the byte of configuration data. The microprocessor can then perform other system functions while the Stratix II or Stratix II GX device is processing the byte of configuration data.

During the time RDYnBSY is low, the Stratix II or Stratix II GX device internally processes the configuration data using its internal oscillator (typically 100 MHz). When the device is ready for the next byte of configuration data, it will drive RDYnBSY high. If the microprocessor senses a high signal when it polls RDYnBSY, the microprocessor sends the next byte of configuration data to the device.

Alternatively, the nRS signal can be strobed low, causing the RDYnBSY signal to appear on DATA7. Because RDYnBSY does not need to be monitored, this pin doesn't need to be connected to the microprocessor. Do not drive data onto the data bus while nRS is low because it will cause contention on the DATA7 pin. If you are not using the nRS pin to monitor configuration, it should be tied high.

To simplify configuration and save an I/O port, the microprocessor can wait for the total time of t_{BUSY} (max) + t_{RDY2WS} + t_{W2SB} before sending the next data byte. In this set-up, nRS should be tied high and RDYnBSY does not need to be connected to the microprocessor. The t_{BUSY}, t_{RDY2WS}, and t_{W2SB} timing specifications are listed in Table 7–18 on page 7–82.

Next, the microprocessor checks nSTATUS and CONF_DONE. If nSTATUS is not low and CONF_DONE is not high, the microprocessor sends the next data byte. However, if nSTATUS is not low and all the configuration data has been received, the device is ready for initialization. The CONF_DONE pin will go high one byte early in parallel configuration (FPP and PPA) modes. The last byte is required for serial configuration (AS and PS) modes. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin. The open-drain CONF_DONE pin is pulled high by an external 10-kΩ pull-up resistor. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

In Stratix II and Stratix II GX devices, the initialization clock source is either the internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Stratix II or Stratix II GX device provides itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device.

You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. The **Enable user-supplied start-up clock** (**CLKUSR**) option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR does not affect the configuration process. After CONF_DONE goes high, CLKUSR is enabled after the time specified as t_{CD2CU}. After this time period elapses, the Stratix II and Stratix II GX devices require 299 clock cycles to initialize properly and enter user mode. Stratix II devices support a CLKUSR f_{MAX} of 100 MHz.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. This **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used it is high because of an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the

INIT_DONE pin is released and pulled high. The microprocessor must be able to detect this low-to-high transition that signals the device has entered user mode. When initialization is complete, the device enters user mode. In user-mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

To ensure DATA[7..0] is not left floating at the end of configuration, the microprocessor must drive them either high or low, whichever is convenient on your board. After configuration, the nCS, CS, nRS, nWS, RDYnBSY, and DATA[7..0] pins can be used as user I/O pins. When choosing the PPA scheme in the Quartus II software as a default, these I/O pins are tri-stated in user mode and should be driven by the microprocessor. To change this default option in the Quartus II software, select the **Dual-Purpose Pins** tab of the **Device & Pin Options** dialog box.

If an error occurs during configuration, the device drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the microprocessor that there is an error. If the **Auto-restart configuration after error** option-available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box-is turned on, the device releases nSTATUS after a reset time-out period (maximum of 100 µs). After nSTATUS is released and pulled high by a pull-up resistor, the microprocessor can try to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

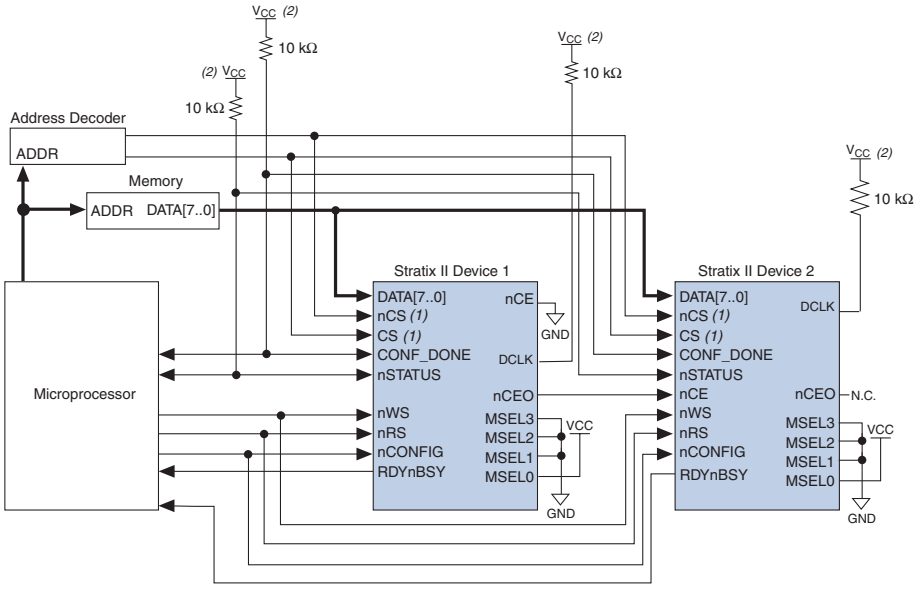
The microprocessor can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. To detect errors and determine when programming completes, monitor the CONF_DONE pin with the microprocessor. If the microprocessor sends all configuration data but CONF_DONE or INIT_DONE has not gone high, the microprocessor must reconfigure the target device.

If you are using the optional CLKUSR pin and nCONFIG is pulled low to restart configuration during device initialization, ensure CLKUSR continues toggling during the time nSTATUS is low (maximum of 100 µs).

When the device is in user-mode, a reconfiguration can be initiated by transitioning the nCONFIG pin low-to-high. The nCONFIG pin should go low for at least 2 µs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Once nCONFIG returns to a logic high level and nSTATUS is released by the device, reconfiguration begins.

Figure 7–31 shows how to configure multiple Stratix II or Stratix II GX devices using a microprocessor. This circuit is similar to the PPA configuration circuit for a single device, except the devices are cascaded for multi-device configuration.

Figure 7–31. Multi-Device PPA Configuration Using a Microprocessor



Notes to Figure 7–31:

(1) If not used, the CS pin can be connected to V_{CC} directly. If not used, the nCS pin can be connected to GND directly.

(2) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain. V_{CC} should be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

In multi-device PPA configuration the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration within one clock cycle. Therefore, the transfer of data destinations is transparent to the microprocessor.

Each device's RDYnBSY pin can have a separate input to the microprocessor. Alternatively, if the microprocessor is pin limited, all the RDYnBSY pins can feed an AND gate and the output of the AND gate can feed the microprocessor. For example, if you have two devices in a PPA

configuration chain, the second device's RDYnBSY pin will be high during the time that the first device is being configured. When the first device has been successfully configured, it will drive nCEO low to activate the next device in the chain and drive its RDYnBSY pin high. Therefore, since RDYnBSY signal is driven high before configuration and after configuration before entering user-mode, the device being configured will govern the output of the AND gate.

The nRS signal can be used in multi-device PPA chain because the Stratix II and Stratix II GX devices tri-state the DATA[7..0] pins before configuration and after configuration before entering user-mode to avoid contention. Therefore, only the device that is currently being configured responds to the nRS strobe by asserting DATA7.

All other configuration pins (nCONFIG, nSTATUS, DATA[7..0], nCS, CS, nWS, nRS and CONF_DONE) are connected to every device in the chain. It is not necessary to tie nCS and CS together for every device in the chain, as each device's nCS and CS input can be driven by a separate source. Configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

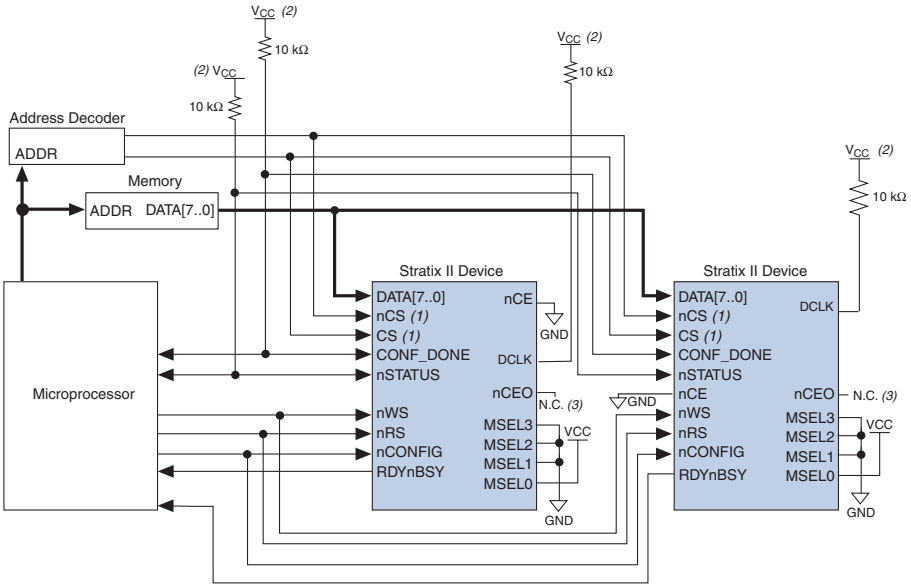
Since all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

If the **Auto-restart configuration after error** option is turned on, the devices release their nSTATUS pins after a reset time-out period (maximum of 100 µs). After all nSTATUS pins are released and pulled high, the microprocessor can try to reconfigure the chain without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

In your system, you may have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DATA[7..0], nCS, CS, nWS, nRS and CONF_DONE) are connected to every device in the chain. Configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DATA lines are buffered for every fourth device. Devices must be the same density and package.

All devices start and complete configuration at the same time. Figure 7–32 shows multi-device PPA configuration when both devices are receiving the same configuration data.

Figure 7–32. Multiple-Device PPA Configuration Using a Microprocessor When Both devices Receive the Same Data



Notes to Figure 7–32:

(1) If not used, the CS pin can be connected to V_{CC} directly. If not used, the nCS pin can be connected to GND directly.

(2) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain. V_{CC} should be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

(3) The nCEO pins of both devices are left unconnected when configuring the same configuration data into multiple devices.

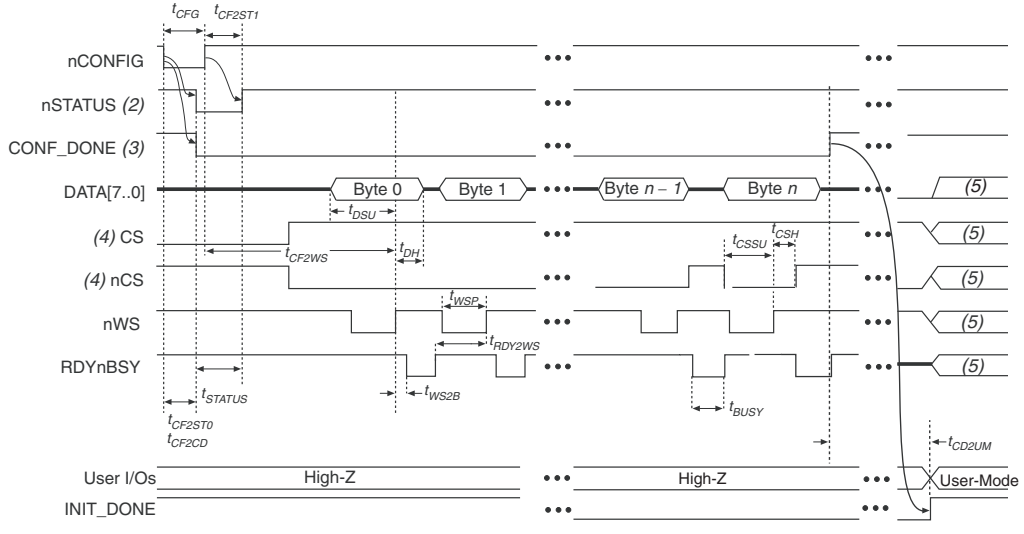
You can use a single configuration chain to configure Stratix II or Stratix II GX devices with other Altera devices that support PPA configuration, such as Stratix, Mercury™, APEX™ 20K, ACEX® 1K, and FLEX® 10KE devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information on configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook*.

PPA Configuration Timing

Figure 7–33 shows the timing waveform for the PPA configuration scheme using a microprocessor.

Figure 7–33. Stratix II and Stratix II GX PPA Configuration Timing Waveform Using nWS *Note (1)*



Notes to Figure 7–33:

(1) The beginning of this waveform shows the device in user-mode. In user-mode, nCONFIG, nSTATUS and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.

(2) Upon power-up, Stratix II and Stratix II GX devices hold nSTATUS low for the time of the POR delay.

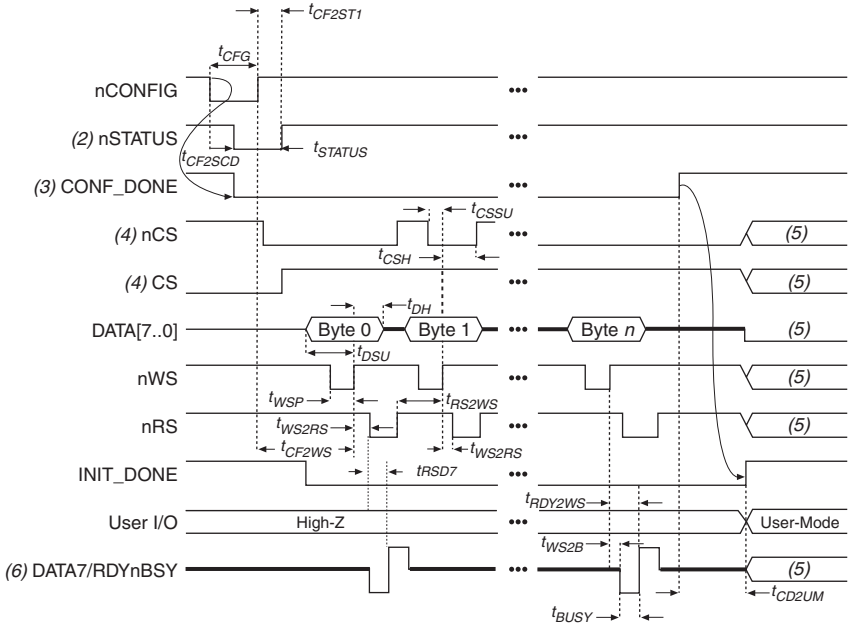
(3) Upon power-up, before and during configuration, CONF_DONE is low.

(4) The user can toggle nCS or CS during configuration if the design meets the specification for t_{CSSU}, t_{WSP}, and t_{CSH}.

(5) DATA[7..0], CS, nCS, nWS, nRS, and RDYnBSY are available as user I/O pins after configuration and the state of theses pins depends on the dual-purpose pin settings.

Figure 7–34 shows the timing waveform for the PPA configuration scheme when using a strobed nRS and nWS signal.

Figure 7–34. Stratix II and Stratix II GX PPA Configuration Timing Waveform Using nRS and nWS *Note (1)*



Notes to Figure 7–34:

(1) The beginning of this waveform shows the device in user-mode. In user-mode, nCONFIG, nSTATUS and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.
(2) Upon power-up, Stratix II and Stratix II GX devices hold nSTATUS low for the time of the POR delay.
(3) Upon power-up, before and during configuration, CONF_DONE is low.
(4) The user can toggle nCS or CS during configuration if the design meets the specification for t_{CSSU}, t_{WSP}, and t_{CSH}.
(5) DATA[7..0], CS, nCS, nWS, nRS, and RDYnBSY are available as user I/O pins after configuration and the state of theses pins depends on the dual-purpose pin settings.
(6) DATA7 is a bidirectional pin. It is an input for configuration data input, but it is an output to show the status of RDYnBSY.

Table 7–18 defines the timing parameters for Stratix II and Stratix II GX devices for PPA configuration.

Table 7–18. PPA Timing Parameters for Stratix II and Stratix II GX Devices (Part 1 of 2)

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		800	ns

Table 7–18. PPA Timing Parameters for Stratix II and Stratix II GX Devices (Part 2 of 2)

Symbol	Parameter	Min	Max	Units
t_{CFG}	nCONFIG low pulse width	2		µs
t_{STATUS}	nSTATUS low pulse width	10	100 *(1)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high		100 *(1)*	µs
t_{CSSU}	Chip select setup time before rising edge on nWS	10		ns
t_{CSH}	Chip select hold time after rising edge on nWS	0		ns
t_{CF2WS}	nCONFIG high to first rising edge on nWS	100		µs
t_{ST2WS}	nSTATUS high to first rising edge on nWS	2		µs
t_{DSU}	Data setup time before rising edge on nWS	10		ns
t_{DH}	Data hold time after rising edge on nWS	0		ns
t_{WSP}	nWS low pulse width	15		ns
t_{WS2B}	nWS rising edge to RDYnBSY low		20	ns
t_{BUSY}	RDYnBSY low pulse width	7	45	ns
t_{RDY2WS}	RDYnBSY rising edge to nWS rising edge	15		ns
t_{WS2RS}	nWS rising edge to nRS falling edge	15		ns
t_{RS2WS}	nRS rising edge to nWS rising edge	15		ns
t_{RSD7}	nRS falling edge to DATA7 valid with RDYnBSY signal		20	ns
t_R	Input rise time		40	ns
t_F	Input fall time		40	ns
t_{CD2UM}	CONF_DONE high to user mode *(2)*	20	100	µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	40		ns
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	t_{CD2CU} + (299 × CLKUSR period)		

Notes to Table 7–18:

(1) This value is obtainable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(2) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device.

Device configuration options and how to create configuration files are discussed further in the *Software Settings* chapter in volume 2 the *Configuration Handbook*.

JTAG Configuration

The JTAG has developed a specification for boundary-scan testing. This boundary-scan test (BST) architecture offers the capability to efficiently test components on PCBs with tight lead spacing. The BST architecture can test pin connections without using physical test probes and capture functional data while a device is operating normally. The JTAG circuitry can also be used to shift configuration data into the device. The Quartus II software automatically generates SOFs that can be used for JTAG configuration with a download cable in the Quartus II software programmer.

For more information on JTAG boundary-scan testing, refer to the following documents:

- *IEEE 1149.1 (JTAG) Boundary-Scan Testing in Stratix II & Stratix II GX Devices* chapter in volume 2 of the *Stratix II Device Handbook* or the *IEEE 1149.1 (JTAG) Boundary-Scan Testing in Stratix II & Stratix II GX Devices* chapter in volume 2 of the *Stratix II GX Device Handbook*
- *Jam Programming Support - JTAG Technologies*

Stratix II and Stratix II GX devices are designed such that JTAG instructions have precedence over any device configuration modes. Therefore, JTAG configuration can take place without waiting for other configuration modes to complete. For example, if you attempt JTAG configuration of Stratix II or Stratix II GX devices during PS configuration, PS configuration is terminated and JTAG configuration begins.

You cannot use the Stratix II and Stratix II GX decompression or design security features if you are configuring your Stratix II or Stratix II GX device when using JTAG-based configuration.

A device operating in JTAG mode uses four required pins, `TDI`, `TDO`, `TMS`, and `TCK`, and one optional pin, `TRST`. The `TCK` pin has an internal weak pull-down resistor, while the `TDI`, `TMS`, and `TRST` pins have weak internal pull-up resistors (typically 25 kΩ). The `TDO` output pin is powered by V_{CCIO} in I/O bank 4. All of the JTAG input pins are powered by the 3.3-V V_{CCPD} pin. All user I/O pins are tri-stated during JTAG configuration. Table 7–19 explains each JTAG pin's function.

The `TDO` output is powered by the V_{CCIO} power supply of I/O bank 4.

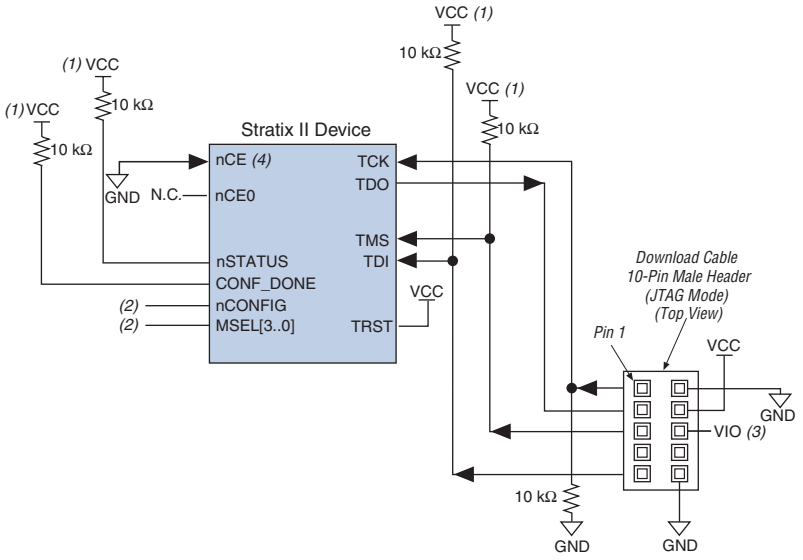
For recommendations on how to connect a JTAG chain with multiple voltages across the devices in the chain, refer to the *IEEE 1149.1 (JTAG) Boundary-Scan Testing in Stratix II & Stratix II GX Devices* chapter in volume 2 of the *Stratix II Device Handbook* or the *IEEE 1149.1 (JTAG) Boundary-Scan Testing in Stratix II & Stratix II GX Devices* chapter in volume 2 of the *Stratix II GX Device Handbook*.

Table 7–19. Dedicated JTAG Pins

Pin Name	Pin Type	Description
TDI	Test data input	Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}.
TDO	Test data output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. If the JTAG circuitry is not used, leave the TDO pin unconnected.
TMS	Test mode select	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}.
TCK	Test clock input	The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND.
TRST	Test reset input (optional)	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to IEEE Std. 1149.1. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND.

During JTAG configuration, data can be downloaded to the device on the PCB through the USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV download cable. Configuring devices through a cable is similar to programming devices in-system, except the TRST pin should be connected to V_{CC}. This ensures that the TAP controller is not reset. Figure 7–35 shows JTAG configuration of a single Stratix II or Stratix II GX device.

Figure 7–35. JTAG Configuration of a Single Device Using a Download Cable



Notes to Figure 7–35:

(1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster (V_{IO} pin), ByteBlaster II, or ByteBlasterMV cable.

(2) The `nCONFIG`, `MSEL[3..0]` pins should be connected to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect `nCONFIG` to V_{CC}, and `MSEL[3..0]` to ground. Pull `DCLK` either high or low, whichever is convenient on your board.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. Refer to the *MasterBlaster Serial/USB Communications Cable User Guide* for this value. In the ByteBlasterMV cable, this pin is a no connect. In the USB Blaster and ByteBlaster II cables, this pin is connected to `nCE` when it is used for active serial programming, otherwise it is a no connect.

(4) `nCE` must be connected to GND or driven low for successful JTAG configuration.

To configure a single device in a JTAG chain, the programming software places all other devices in bypass mode. In bypass mode, devices pass programming data from the `TDI` pin to the `TDO` pin through a single bypass register without being affected internally. This scheme enables the programming software to program or verify the target device. Configuration data driven into the device appears on the `TDO` pin one clock cycle later.

The Quartus II software verifies successful JTAG configuration upon completion. At the end of configuration, the software checks the state of `CONF_DONE` through the JTAG port. When Quartus II generates a (**.jam**) file for a multi-device chain, it contains instructions so that all the devices in the chain will be initialized at the same time. If `CONF_DONE` is not high, the Quartus II software indicates that configuration has failed. If

CONF_DONE is high, the software indicates that configuration was successful. After the configuration bit stream is transmitted serially via the JTAG TDI port, the TCK port is clocked an additional 299 cycles to perform device initialization.

Stratix II and Stratix II GX devices have dedicated JTAG pins that always function as JTAG pins. Not only can you perform JTAG testing on Stratix II and Stratix II GX devices before and after, but also during configuration. While other device families do not support JTAG testing during configuration, Stratix II and Stratix II GX devices support the bypass, idcode, and sample instructions during configuration without interrupting configuration. All other JTAG instructions may only be issued by first interrupting configuration and reprogramming I/O pins using the CONFIG_IO instruction.

The CONFIG_IO instruction allows I/O buffers to be configured via the JTAG port and when issued, interrupts configuration. This instruction allows you to perform board-level testing prior to configuring the Stratix II or Stratix II GX device or waiting for a configuration device to complete configuration. Once configuration has been interrupted and JTAG testing is complete, the part must be reconfigured via JTAG (PULSE_CONFIG instruction) or by pulsing nCONFIG low.

The chip-wide reset (DEV_CLRn) and chip-wide output enable (DEV_OE) pins on Stratix II and Stratix II GX devices do not affect JTAG boundary-scan or programming operations. Toggling these pins does not affect JTAG operations (other than the usual boundary-scan operation).

When designing a board for JTAG configuration of Stratix II or Stratix II GX devices, consider the dedicated configuration pins. Table 7–20 shows how these pins should be connected during JTAG configuration.

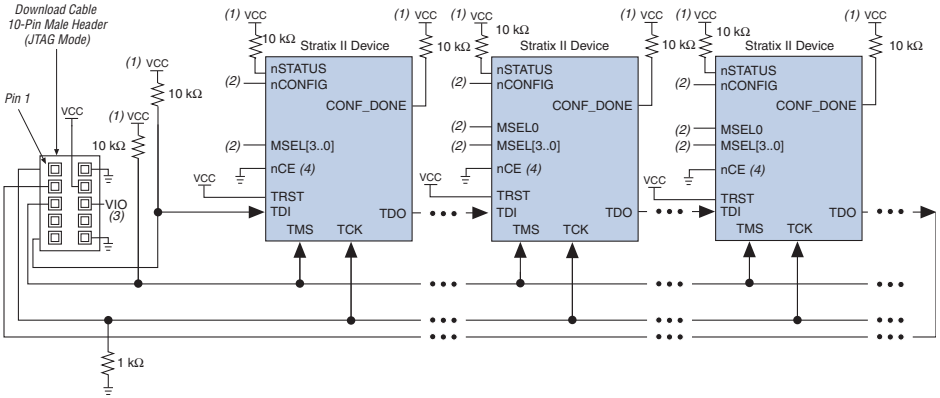
When programming a JTAG device chain, one JTAG-compatible header is connected to several devices. The number of devices in the JTAG chain is limited only by the drive capability of the download cable. When four or more devices are connected in a JTAG chain, Altera recommends buffering the TCK, TDI, and TMS pins with an on-board buffer.

Table 7–20. Dedicated Configuration Pin Connections During JTAG Configuration

Signal	Description
nCE	On all Stratix II or Stratix II GX devices in the chain, nCE should be driven low by connecting it to ground, pulling it low via a resistor, or driving it by some control circuitry. For devices that are also in multi-device FPP, AS, PS, or PPA configuration chains, the nCE pins should be connected to GND during JTAG configuration or JTAG configured in the same order as the configuration chain.
nCEO	On all Stratix II or Stratix II GX devices in the chain, nCEO can be left floating or connected to the nCE of the next device.
MSEL	These pins must not be left floating. These pins support whichever non-JTAG configuration is used in production. If only JTAG configuration is used, tie these pins to ground.
nCONFIG	Driven high by connecting to V_{CC}, pulling up via a resistor, or driven high by some control circuitry.
nSTATUS	Pull to V_{CC} via a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each nSTATUS pin should be pulled up to V_{CC} individually.
CONF_DONE	Pull to V_{CC} via a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each CONF_DONE pin should be pulled up to V_{CC} individually. CONF_DONE going high at the end of JTAG configuration indicates successful configuration.
DCLK	Should not be left floating. Drive low or high, whichever is more convenient on your board.

JTAG-chain device programming is ideal when the system contains multiple devices, or when testing your system using JTAG BST circuitry. Figure 7–36 shows multi-device JTAG configuration.

Figure 7–36. JTAG Configuration of Multiple Devices Using a Download Cable



Notes to Figure 7–36:

(1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster (V_{IO} pin), ByteBlaster II or ByteBlasterMV cable.

(2) The nCONFIG, MSEL[3..0] pins should be connected to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect nCONFIG to V_{CC}, and MSEL[3..0] to ground. Pull DCLK either high or low, whichever is convenient on your board.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. Refer to the *MasterBlaster Serial/USB Communications Cable User Guide* for this value. In the ByteBlasterMV cable, this pin is a no connect. In the USB Blaster and ByteBlaster II cables, this pin is connected to nCE when it is used for active serial programming, otherwise it is a no connect.

(4) nCE must be connected to GND or driven low for successful JTAG configuration.

The nCE pin must be connected to GND or driven low during JTAG configuration. In multi-device FPP, AS, PS, and PPA configuration chains, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. In addition, the CONF_DONE and nSTATUS signals are all shared in multi-device FPP, AS, PS, or PPA configuration chains so the devices can enter user mode at the same time after configuration is complete. When the CONF_DONE and nSTATUS signals are shared among all the devices, every device must be configured when JTAG configuration is performed.

If you only use JTAG configuration, Altera recommends that you connect the circuitry as shown in Figure 7–36, where each of the CONF_DONE and nSTATUS signals are isolated, so that each device can enter user mode individually.

After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. Therefore, if these devices are also in a JTAG chain, make sure the nCE pins are connected to GND during JTAG configuration or that the devices are JTAG configured in the same order as the configuration chain. As long as the devices are JTAG configured in the same order as the multi-device configuration chain, the nCEO of the previous device will drive nCE of the next device low when it has successfully been JTAG configured.

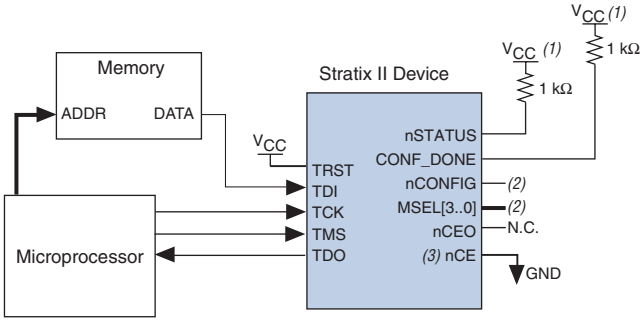
Other Altera devices that have JTAG support can be placed in the same JTAG chain for device programming and configuration.

Stratix, Stratix II, Stratix II GX, Cyclone™, and Cyclone II devices must be within the first 17 devices in a JTAG chain. All of these devices have the same JTAG controller. If any of the Stratix, Stratix II, Stratix II GX, Cyclone, and Cyclone II devices are in the 18th or after they will fail configuration. This does not affect SignalTap® II.

For more information on configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in the *Configuration Handbook*.

Figure 7–37 shows JTAG configuration of a Stratix II or Stratix II GX device with a microprocessor.

Figure 7–37. JTAG Configuration of a Single Device Using a Microprocessor



Notes to Figure 7–37:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain. V_{CC} should be high enough to meet the V_{IH} specification of the I/O on the device.

(2) The `nCONFIG`, `MSEL[3..0]` pins should be connected to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect `nCONFIG` to V_{CC}, and `MSEL[3..0]` to ground. Pull `DCLK` either high or low, whichever is convenient on your board.

(3) `nCE` must be connected to GND or driven low for successful JTAG configuration.

Jam STAPL

Jam STAPL, JEDEC standard JESD-71, is a standard file format for in-system programmability (ISP) purposes. Jam STAPL supports programming or configuration of programmable devices and testing of electronic systems, using the IEEE 1149.1 JTAG interface. Jam STAPL is a freely licensed open standard.

The Jam Player provides an interface for manipulating the IEEE Std. 1149.1 JTAG TAP state machine.

For more information on JTAG and Jam STAPL in embedded environments, refer to *AN 122: Using Jam STAPL for ISP & ICR via an Embedded Processor*. To download the jam player, visit the Altera web site at www.altera.com.

Device Configuration Pins

The following tables describe the connections and functionality of all the configuration related pins on the Stratix II and Stratix II GX devices. Table 7–21 summarizes the Stratix II pin configuration.

Table 7–21. Stratix II Configuration Pin Summary (Part 1 of 2) *Note (1)*

Bank	Description	Input/Output	Dedicated	Powered By	Configuration Mode
3	PGM[2..0]	Output		*(2)*	PS, FPP, PPA, RU, LU
3	ASDO	Output		*(2)*	AS
3	nCSO	Output		*(2)*	AS
3	CRC_ERROR	Output		*(2)*	Optional, all modes
3	DATA0	Input		*(3)*	All modes except JTAG
3	DATA[7..1]	Input		*(3)*	FPP, PPA
3	DATA7	Bidirectional		*(2)*, *(3)*	PPA
3	RDYnBSY	Output		*(2)*	PPA
3	INIT_DONE	Output		Pull-up	Optional, all modes
3	nSTATUS	Bidirectional	Yes	Pull-up	All modes
3	nCE	Input	Yes	*(3)*	All modes
3	DCLK	Input	Yes	*(3)*	PS, FPP
		Output		*(2)*	AS
3	CONF_DONE	Bidirectional	Yes	Pull-up	All modes
8	TDI	Input	Yes	VCCPD	JTAG
8	TMS	Input	Yes	VCCPD	JTAG
8	TCK	Input	Yes	VCCPD	JTAG
8	TRST	Input	Yes	VCCPD	JTAG
8	nCONFIG	Input	Yes	*(3)*	All modes
8	VCCSEL	Input	Yes	VCCINT	All modes
8	CS	Input		*(3)*	PPA
8	CLKUSR	Input		*(3)*	Optional
8	nWS	Input		*(3)*	PPA
8	nRS	Input		*(3)*	PPA
8	RUnLU	Input		*(3)*	PS, FPP, PPA, RU, LU
8	nCS	Input		*(3)*	PPA
7	PORSEL	Input	Yes	VCCINT	All modes
7	nIO_PULLUP	Input	Yes	VCCINT	All modes

Table 7–21. Stratix II Configuration Pin Summary (Part 2 of 2) *Note (1)*

Bank	Description	Input/Output	Dedicated	Powered By	Configuration Mode
7	PLL_ENA	Input	Yes	*(3)*	Optional
7	nCEO	Output	Yes	*(2)*, *(4)*	All modes
4	MSEL[3..0]	Input	Yes	VCCINT	All modes
4	TDO	Output	Yes	*(2)*, *(4)*	JTAG

Notes to Table 7–21:

(1) Total number of pins is 41, total number of dedicated pins is 19.

(2) All outputs are powered by VCCIO except as noted.

(3) All inputs are powered by VCCIO or VCCPD, based on the VCCSEL setting, except as noted.

(4) An external pull-up resistor may be required for this configuration pin because of the multivolt I/O interface. Refer to the *Stratix II Architecture* chapter in volume 1 of the *Stratix II Device Handbook* for pull-up or level shifter recommendations for nCEO and TDO.

Figure 7–38 shows the I/O bank locations.

Figure 7–38. Stratix II I/O Bank Numbers

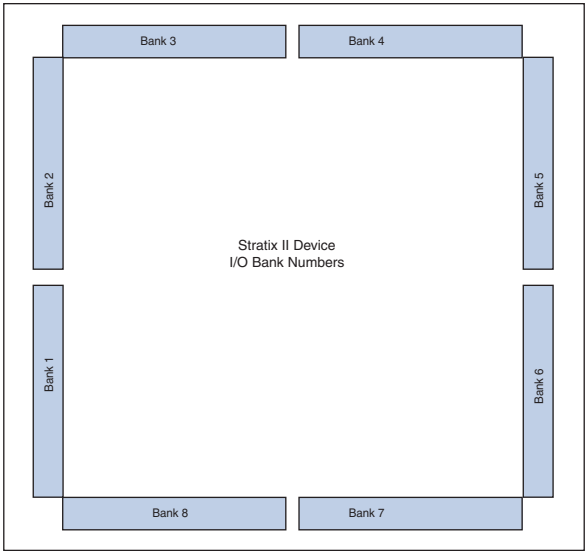


Table 7–22 describes the dedicated configuration pins, which are required to be connected properly on your board for successful configuration. Some of these pins may not be required for your configuration schemes.

Table 7–22. Dedicated Configuration Pins on the Stratix II and Stratix II GX Device (Part 1 of 10)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
V_{CCPD}	N/A	All	Power	Dedicated power pin. This pin is used to power the I/O pre-drivers, the JTAG input pins, and the configuration input pins that are affected by the voltage level of VCCSEL. This pin must be connected to 3.3-V. V_{CCPD} must ramp-up from 0-V to 3.3-V within 100 ms. If V_{CCPD} is not ramped up within this specified time, your Stratix II or Stratix II GX device will not configure successfully. If your system does not allow for a V_{CCPD} ramp-up time of 100 ms or less, you must hold nCONFIG low until all power supplies are stable.
VCCSEL	N/A	All	Input	Dedicated input that selects which input buffer is used on the PLL_ENA pin and the configuration input pins; nCONFIG, DCLK (when used as an input), nSTATUS (when used as an input), CONF_DONE (when used as an input), DEV_OE, DEV_CLRn, DATA[7..0], RUnLU, nCE, nWS, nRS, CS, nCS, and CLKUSR. The 3.3-V/2.5-V input buffer is powered by V_{CCPD}, while the 1.8-V/1.5-V input buffer is powered by V_{CCIO}. The VCCSEL input buffer has an internal 5-kΩ pull-down resistor that is always active. The VCCSEL input buffer is powered by V_{CCINT} and must be hardwired to V_{CCPD} or ground. A logic high selects the 1.8-V/1.5-V input buffer, and a logic low selects the 3.3-V/2.5-V input buffer. For more information, refer to the "VCCSEL Pin" section.

Table 7–22. Dedicated Configuration Pins on the Stratix II and Stratix II GX Device (Part 2 of 10)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
PORSEL	N/A	All	Input	Dedicated input which selects between a POR time of 12 ms or 100 ms. A logic high (1.5 V, 1.8 V, 2.5 V, 3.3 V) selects a POR time of about 12 ms and a logic low selects POR time of about 100 ms. The PORSEL input buffer is powered by V_{CCINT} and has an internal 5-kΩ pull-down resistor that is always active. The PORSEL pin should be tied directly to V_{CCPD} or GND.
nIO_PULLUP	N/A	All	Input	Dedicated input that chooses whether the internal pull-up resistors on the user I/O pins and dual-purpose I/O pins (nCS0, nASD0, DATA[7..0], nWS, nRS, RDYnBSY, nCS, CS, RUnLU, PGM[], CLKUSR, INIT_DONE, DEV_OE, DEV_CLR) are on or off before and during configuration. A logic high (1.5 V, 1.8 V, 2.5 V, 3.3 V) turns off the weak internal pull-up resistors, while a logic low turns them on. The nIO-PULLUP input buffer is powered by V_{CCPD} and has an internal 5-kΩ pull-down resistor that is always active. The nIO-PULLUP can be tied directly to V_{CCPD} or use a 1-kΩ pull-up resistor or tied directly to GND.
MSEL[3..0]	N/A	All	Input	4-bit configuration input that sets the Stratix II and Stratix II GX device configuration scheme. Refer to Table 7–1 for the appropriate connections. These pins must be hard-wired to V_{CCPD} or GND. The MSEL[3..0] pins have internal 5-kΩ pull-down resistors that are always active.

Table 7–22. Dedicated Configuration Pins on the Stratix II and Stratix II GX Device (Part 3 of 10)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nCONFIG	N/A	All	Input	Configuration control input. Pulling this pin low during user-mode will cause the device to lose its configuration data, enter a reset state, tri-state all I/O pins. Returning this pin to a logic high level will initiate a reconfiguration. If your configuration scheme uses an enhanced configuration device or EPC2 device, nCONFIG can be tied directly to V_{CC} or to the configuration device's nINIT_CONF pin.
nSTATUS	N/A	All	Bidirectional open-drain	The device drives nSTATUS low immediately after power-up and releases it after the POR time. Status output. If an error occurs during configuration, nSTATUS is pulled low by the target device. Status input. If an external source drives the nSTATUS pin low during configuration or initialization, the target device enters an error state. Driving nSTATUS low after configuration and initialization does not affect the configured device. If a configuration device is used, driving nSTATUS low will cause the configuration device to attempt to configure the device, but since the device ignores transitions on nSTATUS in user-mode, the device does not reconfigure. To initiate a reconfiguration, nCONFIG must be pulled low. The enhanced configuration devices' and EPC2 devices' OE and nCS pins have optional internal programmable pull-up resistors. If internal pull-up resistors on the enhanced configuration device are used, external 10-kΩ pull-up resistors should not be used on these pins. When using EPC2 devices, only external 10-kΩ pull-up resistors should be used.

Table 7–22. Dedicated Configuration Pins on the Stratix II and Stratix II GX Device (Part 4 of 10)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nSTATUS (continued)				If VCCPD and VCCIO are not fully powered up, the following could occur: • VCCPD and VCCIO are powered high enough for the nSTATUS buffer to function properly, and nSTATUS is driven low. When VCCPD and VCCIO are ramped up, POR trips, and nSTATUS is released after POR expires. • VCCPD and VCCIO are not powered high enough for the nSTATUS buffer to function properly. In this situation, nSTATUS might appear logic high, triggering a configuration attempt that would fail because POR did not yet trip. When VCCPD and VCCIO are powered up, nSTATUS is pulled low because POR did not yet trip. When POR trips after VCCPD and VCCIO are powered up, nSTATUS is released and pulled high. At that point, reconfiguration is triggered and the device is configured.

Table 7–22. Dedicated Configuration Pins on the Stratix II and Stratix II GX Device (Part 5 of 10)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
CONF_DONE	N/A	All	Bidirectional open-drain	Status output. The target device drives the CONF_DONE pin low before and during configuration. Once all configuration data is received without error and the initialization cycle starts, the target device releases CONF_DONE. Status input. After all data is received and CONF_DONE goes high, the target device initializes and enters user mode. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize. Driving CONF_DONE low after configuration and initialization does not affect the configured device. The enhanced configuration devices' and EPC2 devices' OE and nCS pins have optional internal programmable pull-up resistors. If internal pull-up resistors on the enhanced configuration device are used, external 10-kΩ pull-up resistors should not be used on these pins. When using EPC2 devices, only external 10-kΩ pull-up resistors should be used.
nCE	N/A	All	Input	Active-low chip enable. The nCE pin activates the device with a low signal to allow configuration. The nCE pin must be held low during configuration, initialization, and user mode. In single device configuration, it should be tied low. In multi-device configuration, nCE of the first device is tied low while its nCEO pin is connected to nCE of the next device in the chain. The nCE pin must also be held low for successful JTAG programming of the device.

Table 7–22. Dedicated Configuration Pins on the Stratix II and Stratix II GX Device (Part 6 of 10)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nCE0	N/A	All	Output	Output that drives low when device configuration is complete. In single device configuration, this pin is left floating. In multi-device configuration, this pin feeds the next device's nCE pin. The nCE0 of the last device in the chain is left floating. The nCE0 pin is powered by V_{CCIO} in I/O bank 7. For recommendations on how to connect nCE0 in a chain with multiple voltages across the devices in the chain, refer to the *Stratix II Architecture* chapter in volume 1 of the *Stratix II Handbook* or the *Stratix II GX Architecture* chapter in volume 1 of the *Stratix II GX Device Handbook*.
ASD0	N/A in AS mode I/O in non-AS mode	AS	Output	Control signal from the Stratix II or Stratix II GX device to the serial configuration device in AS mode used to read out configuration data. In AS mode, ASD0 has an internal pull-up resistor that is always active.
nCS0	N/A in AS mode I/O in non-AS mode	AS	Output	Output control signal from the Stratix II or Stratix II GX device to the serial configuration device in AS mode that enables the configuration device. In AS mode, nCS0 has an internal pull-up resistor that is always active.

Table 7–22. Dedicated Configuration Pins on the Stratix II and Stratix II GX Device (Part 7 of 10)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
DCLK	N/A	Synchronous configuration schemes (PS, FPP, AS)	Input (PS, FPP) Output (AS)	In PS and FPP configuration, DCLK is the clock input used to clock data from an external source into the target device. Data is latched into the device on the rising edge of DCLK. In AS mode, DCLK is an output from the Stratix II or Stratix II GX device that provides timing for the configuration interface. In AS mode, DCLK has an internal pull-up resistor (typically 25 kΩ) that is always active. In PPA mode, DCLK should be tied high to V_{CC} to prevent this pin from floating. After configuration, this pin is tri-stated. In schemes that use a configuration device, DCLK will be driven low after configuration is done. In schemes that use a control host, DCLK should be driven either high or low, whichever is more convenient. Toggling this pin after configuration does not affect the configured device.
DATA0	I/O	PS, FPP, PPA, AS	Input	Data input. In serial configuration modes, bit-wide configuration data is presented to the target device on the DATA0 pin. The V_{IH} and V_{IL} levels for this pin are dependent on the input buffer selected by the VCCSEL pin. Refer to the section "VCCSEL Pin" on page 7–10 for more information. In AS mode, DATA0 has an internal pull-up resistor that is always active. After configuration, DATA0 is available as a user I/O pin and the state of this pin depends on the **Dual-Purpose Pin** settings. After configuration, EPC1 and EPC1441 devices tri-state this pin, while enhanced configuration and EPC2 devices drive this pin high.

Table 7–22. Dedicated Configuration Pins on the Stratix II and Stratix II GX Device (Part 8 of 10)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
DATA[7..1]	I/O	Parallel configuration schemes (FPP and PPA)	Inputs	Data inputs. Byte-wide configuration data is presented to the target device on DATA[7..0]. The V_{IH} and V_{IL} levels for this pin are dependent on the input buffer selected by the VCCSEL pin. Refer to the section "VCCSEL Pin" on page 7–10 for more information. In serial configuration schemes, they function as user I/O pins during configuration, which means they are tri-stated. After PPA or FPP configuration, DATA[7..1] are available as user I/O pins and the state of these pin depends on the **Dual-Purpose Pin** settings.
DATA7	I/O	PPA	Bidirectional	In the PPA configuration scheme, the DATA7 pin presents the RDYnBSY signal after the nRS signal has been strobed low. The V_{IH} and V_{IL} levels for this pin are dependent on the input buffer selected by the VCCSEL pin. Refer to the section "VCCSEL Pin" on page 7–10 for more information. In serial configuration schemes, it functions as a user I/O pin during configuration, which means it is tri-stated. After PPA configuration, DATA7 is available as a user I/O and the state of this pin depends on the **Dual-Purpose Pin** settings.
nWS	I/O	PPA	Input	Write strobe input. A low-to-high transition causes the device to latch a byte of data on the DATA[7..0] pins. In non-PPA schemes, it functions as a user I/O pin during configuration, which means it is tri-stated. After PPA configuration, nWS is available as a user I/O pins and the state of this pin depends on the **Dual-Purpose Pin** settings.

Table 7–22. Dedicated Configuration Pins on the Stratix II and Stratix II GX Device (Part 9 of 10)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nRS	I/O	PPA	Input	Read strobe input. A low input directs the device to drive the RDYnBSY signal on the DATA7 pin. If the nRS pin is not used in PPA mode, it should be tied high. In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, nRS is available as a user I/O pin and the state of this pin depends on the **Dual-Purpose Pin** settings.
RDYnBSY	I/O	PPA	Output	Ready output. A high output indicates that the target device is ready to accept another data byte. A low output indicates that the target device is busy and not ready to receive another data byte. In PPA configuration schemes, this pin will drive out high after power-up, before configuration and after configuration before entering user-mode. In non-PPA schemes, it functions as a user I/O pin during configuration, which means it is tri-stated. After PPA configuration, RDYnBSY is available as a user I/O pin and the state of this pin depends on the **Dual-Purpose Pin** settings.

Table 7–22. Dedicated Configuration Pins on the Stratix II and Stratix II GX Device (Part 10 of 10)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nCS/CS	I/O	PPA	Input	Chip-select inputs. A low on nCS and a high on CS select the target device for configuration. The nCS and CS pins must be held active during configuration and initialization. During the PPA configuration mode, it is only required to use either the nCS or CS pin. Therefore, if only one chip-select input is used, the other must be tied to the active state. For example, nCS can be tied to ground while CS is toggled to control configuration. In non-PPA schemes, it functions as a user I/O pin during configuration, which means it is tri-stated. After PPA configuration, nCS and CS are available as user I/O pins and the state of these pins depends on the **Dual-Purpose Pin** settings.
RUnLU	N/A if using Remote System Upgrade I/O if not	Remote System Upgrade in FPP, PS or PPA	Input	Input that selects between remote update and local update. A logic high (1.5-V, 1.8-V, 2.5-V, 3.3-V) selects remote update and a logic low selects local update. When not using remote update or local update configuration modes, this pin is available as general-purpose user I/O pin.
PGM[2..0]	N/A if using Remote System Upgrade I/O if not using	Remote System Upgrade in FPP, PS or PPA	Output	These output pins select one of eight pages in the memory (either flash or enhanced configuration device) when using a remote system upgrade mode. When not using remote update or local update configuration modes, these pins are available as general-purpose user I/O pins.

Table 7–23 describes the optional configuration pins. If these optional configuration pins are not enabled in the Quartus II software, they are available as general-purpose user I/O pins. Therefore, during configuration, these pins function as user I/O pins and are tri-stated with weak pull-up resistors.

Table 7–23. Optional Configuration Pins

Pin Name	User Mode	Pin Type	Description
CLKUSR	N/A if option is on. I/O if option is off.	Input	Optional user-supplied clock input synchronizes the initialization of one or more devices. This pin is enabled by turning on the **Enable user-supplied start-up clock** (**CLKUSR**) option in the Quartus II software.
INIT_DONE	N/A if option is on. I/O if option is off.	Output open-drain	Status pin can be used to indicate when the device has initialized and is in user mode. When nCONFIG is low and during the beginning of configuration, the INIT_DONE pin is tri-stated and pulled high due to an external 10-kΩ pull-up resistor. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low. When initialization is complete, the INIT_DONE pin will be released and pulled high and the device enters user mode. Thus, the monitoring circuitry must be able to detect a low-to-high transition. This pin is enabled by turning on the **Enable INIT_DONE output** option in the Quartus II software.
DEV_OE	N/A if option is on. I/O if option is off.	Input	Optional pin that allows the user to override all tri-states on the device. When this pin is driven low, all I/O pins are tri-stated; when this pin is driven high, all I/O pins behave as programmed. This pin is enabled by turning on the **Enable device-wide output enable** (**DEV_OE**) option in the Quartus II software.
DEV_CLRn	N/A if option is on. I/O if option is off.	Input	Optional pin that allows you to override all clears on all device registers. When this pin is driven low, all registers are cleared; when this pin is driven high, all registers behave as programmed. This pin is enabled by turning on the **Enable device-wide reset** (**DEV_CLRn**) option in the Quartus II software.

Table 7–24 describes the dedicated JTAG pins. JTAG pins must be kept stable before and during configuration to prevent accidental loading of JTAG instructions. The TDI, TMS, and TRST have weak internal pull-up resistors (typically 25 kΩ) while TCK has a weak internal pull-down resistor. If you plan to use the SignalTap embedded logic array analyzer, you need to connect the JTAG pins of the Stratix II or Stratix II GX device to a JTAG header on your board.

Table 7–24. Dedicated JTAG Pins (Part 1 of 2)

Pin Name	User Mode	Pin Type	Description
TDI	N/A	Input	Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. The TDI pin is powered by the 3.3-V V_{CCPD} supply. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}.
TDO	N/A	Output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. The TDO pin is powered by V_{CCIO} in I/O bank 4. For recommendations on connecting a JTAG chain with multiple voltages across the devices in the chain, refer to the *IEEE 1149.1 (JTAG) Boundary Scan Testing in Stratix II & Stratix II GX Devices* chapter in volume 2 of the *Stratix II Handbook* or the *IEEE 1149.1 (JTAG) Boundary Scan Testing in Stratix II & Stratix II GX Devices* chapter in volume 2 of the *Stratix II GX Device Handbook*. If the JTAG circuitry is not used, leave the TDO pin unconnected.
TMS	N/A	Input	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. The TMS pin is powered by the 3.3-V V_{CCPD} supply. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}.

Table 7–24. Dedicated JTAG Pins (Part 2 of 2)

Pin Name	User Mode	Pin Type	Description
TCK	N/A	Input	The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. The TCK pin is powered by the 3.3-V V_{CCPD} supply. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting TCK to GND.
TRST	N/A	Input	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to IEEE Std. 1149.1. The TRST pin is powered by the 3.3-V V_{CCPD} supply. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting the TRST pin to GND.

Conclusion

Stratix II and Stratix II GX devices can be configured in a number of different schemes to fit your system's need. In addition, configuration bitstream encryption, configuration data decompression, and remote system upgrade support supplement the Stratix II and Stratix II GX configuration solution.

Referenced Documents

This chapter references the following documents:

- *AN 122: Using Jam STAPL for ISP & ICR via an Embedded Processor*
- *AN 418: SRunner: An Embedded Solution for Serial Configuration Device Programming*
- *ByteBlaster II Download Cable User Guide*
- *ByteBlasterMV Download Cable User Guide*
- *Configuration Devices for SRAM-Based LUT Devices Data Sheet* chapter in volume 2 of the *Configuration Handbook*.
- *Configuring Mixed Altera FPGA Chains* in volume 2 of the *Configuration Handbook*
- *DC & Switching Characteristics* chapter in volume 1 of the *Stratix II Device Handbook*
- *DC & Switching Characteristics* chapter in volume 1 of the *Stratix II GX Device Handbook*
- *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* in volume 2 of the *Configuration Handbook*
- *IEEE 1149.1 (JTAG) Boundary-Scan Testing in Stratix II & Stratix II GX Devices* chapter in volume 2 of the *Stratix II Device Handbook*
- *IEEE 1149.1 (JTAG) Boundary-Scan Testing in Stratix II & Stratix II GX Devices* chapter in volume 2 of the *Stratix II GX Device Handbook*
- *Jam Programming Support - JTAG Technologies*
- *MasterBlaster Serial/USB Communications Cable User Guide*

- *Remote System Upgrades With Stratix II & Stratix II GX Devices* chapter in volume 2 of the *Stratix II Device Handbook*
- *Remote System Upgrades With Stratix II & Stratix II GX Devices* chapter in volume 2 of the *Stratix II GX Device Handbook*.
- *USB-Blaster Download Cable User Guide*
- *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 2 of the *Configuration Handbook*.
- *Software Settings* in volume 2 of the *Configuration Handbook*
- *Stratix II GX Architecture* chapter in volume 1 of the *Stratix II GX Device Handbook*
- *Stratix II Device Handbook*
- *Stratix II GX Device Handbook*

Document Revision History

Table 7–25 shows the revision history for this chapter.

Table 7–25. Document Revision History (Part 1 of 2)

Date and Document Version	Changes Made	Summary of Changes
January 2008, v4.5	Updated RUnLU and PGM[2..0] information in Table 7–22.	—
	Updated handnote in "Configuration Data Decompression" section.	
	Updated Notes in: • Table 7–3 • Table 7–10 • Table 7–12 • Table 7–15 • Table 7–18	—
	Updated TDO row for Tables 7–19 and 7–24.	—
	Added the "Referenced Documents" section.	—
	Minor text edits.	—
No change	For the *Stratix II GX Device Handbook* only: Formerly chapter 12. The chapter number changed due to the addition of the *Stratix II GX Dynamic Reconfiguration* chapter. No content change.	—

Table 7–25. Document Revision History (Part 2 of 2)

Date and Document Version	Changes Made	Summary of Changes
May 2007, v4.4	• Updated "Power-On Reset Circuit" section • Updated "VCCSEL Pin" section • Updated "Configuration Data Decompression" section • Updated "Active Serial Configuration (Serial Configuration Devices)" section • Updated "Output Configuration Pins" section	—
	Added notes to "FPP Configuration Using a MAX II Device as an External Host" and "Passive Parallel Asynchronous Configuration" section.	—
	• Updated Table 13–5 • Updated Table 13–6 • Updated Table 13–8 • Updated Table 13–11	—
	Removed Table 7-7.	—
	Added new "Output Configuration Pins" section.	—
	Corrected typo in "Configuration Devices" section.	—
	Corrected CONF_DONE in Table 13–22.	—
February 2007 v4.3	Added the "Document Revision History" section to this chapter.	—
April 2006, v4.2	Chapter updated as part of the *Stratix II Device Handbook* update.	—
No change	Formerly chapter 11. Chapter number change only due to chapter addition to Section I in February 2006; no content change.	—
December 2005, v4.1	Chapter updated as part of the *Stratix II Device Handbook* update.	—
October 2005 v4.0	Added chapter to the *Stratix II GX Device Handbook*.	—

11. Configuring Stratix & Stratix GX Devices

S52013-3.2

Introduction

You can configure Stratix® and Stratix GX devices using one of several configuration schemes. All configuration schemes use either a microprocessor, configuration device, or a download cable. See Table 11–1.

Table 11–1. Stratix & Stratix GX Device Configuration Schemes

Configuration Scheme	Typical Use
Fast passive parallel (FPP)	Configuration with a parallel synchronous configuration device or microprocessor interface where eight bits of configuration data are loaded on every clock cycle.
Passive serial (PS)	Configuration with a serial synchronous microprocessor interface or the MasterBlaster™ communications cable, USB Blaster, ByteBlaster™ II, or ByteBlasterMV parallel port download cable.
Passive parallel asynchronous (PPA)	Configuration with a parallel asynchronous microprocessor interface. In this scheme, the microprocessor treats the target device as memory.
Remote/local update FPP	Configuration using a Nios™ (16-bit ISA) and Nios® II (32-bit ISA) or other embedded processor. Allows you to update the Stratix or Stratix GX device configuration remotely using the FPP scheme to load data.
Remote/local update PS	Passive serial synchronous configuration using a Nios or other embedded processor. Allows you to update the Stratix or Stratix GX device configuration remotely using the PS scheme to load data.
Remote/local update PPA	Passive parallel asynchronous configuration using a Nios or other embedded processor. In this scheme, the Nios microprocessor treats the target device as memory. Allows you to update the Stratix or Stratix GX device configuration remotely using the PPA scheme to load data.
Joint Test Action Group (JTAG)	Configuration through the IEEE Std. 1149.1 JTAG pins. You can perform JTAG configuration with either a download cable or an embedded device. Ability to use SignalTap® II Embedded Logic Analyzer.

This chapter discusses how to configure one or more Stratix or Stratix GX devices. It should be used together with the following documents:

- *MasterBlaster Serial/USB Communications Cable Data Sheet*
- *USB Blaster USB Port Download Cable Development Tools Data Sheet*
- *ByteBlaster II Parallel Port Download Cable Data Sheet*
- *ByteBlasterMV Parallel Port Download Cable Data Sheets*
- *Configuration Devices for SRAM-Based LUT Devices Data Sheet*
- *Enhanced Configuration Devices (EPC4, EPC8, & EPC16) Data Sheet*

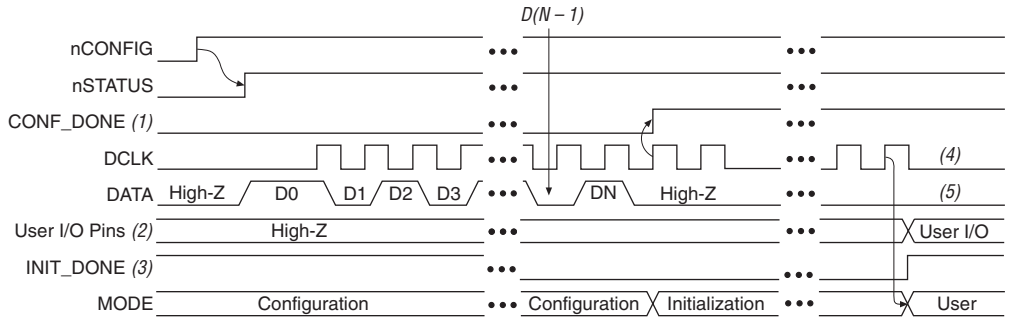
- The *Remote System Configuration with Stratix & Stratix GX Devices* chapter

For more information on setting device configuration options or generating configuration files, see the *Software Setting* chapter in *Volume 2* of the *Configuration Handbook*.

Device Configuration Overview

During device operation, the FPGA stores configuration data in SRAM cells. Because SRAM memory is volatile, you must load the SRAM cells with the configuration data each time the device powers up. After configuration, the device must initialize its registers and I/O pins. After initialization, the device enters user mode. Figure 11–1 shows the state of the device during the configuration, initialization, and user mode.

Figure 11–1. Stratix & Stratix GX Configuration Cycle



Notes to Figure 11–1:

(1) During initial power up and configuration, CONF_DONE is low. After configuration, CONF_DONE goes high. If the device is reconfigured, CONF_DONE goes low after nCONFIG is driven low.

(2) User I/O pins are tri-stated during configuration. Stratix and Stratix GX devices also have a weak pull-up resistor on I/O pins during configuration that are enabled by nIO_PULLUP. After initialization, the user I/O pins perform the function assigned in the user's design.

(3) If the INIT_DONE pin is used, it will be high because of an external 10 kΩ resistor pull-up when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low.

(4) DCLK should not be left floating. It should be driven high or low.

(5) DATA0 should not be left floating. It should be driven high or low.

You can load the configuration data for the Stratix or Stratix GX device using a passive configuration scheme. When using any passive configuration scheme, the Stratix or Stratix GX device is incorporated into a system with an intelligent host, such as a microprocessor, that controls the configuration process. The host supplies configuration data from a storage device (e.g., a hard disk, RAM, or other system memory). When using passive configuration, you can change the target device's

functionality while the system is in operation by reconfiguring the device. You can also perform in-field upgrades by distributing a new programming file to system users.

The following sections describe the MSEL[2..0], VCCSEL, PORSEL, and nIO_PULLUP pins used in Stratix and Stratix GX device configuration.

MSEL[2..0] Pins

You can select a Stratix or Stratix GX device configuration scheme by driving its MSEL2, MSEL1, and MSEL0 pins either high or low, as shown in Table 11–2.

Table 11–2. Stratix & Stratix GX Device Configuration Schemes

Description	MSEL2	MSEL1	MSEL0
FPP configuration	0	0	0
PPA configuration	0	0	1
PS configuration	0	1	0
Remote/local update FPP *(1)*	1	0	0
Remote/local update PPA *(1)*	1	0	1
Remote/local update PS *(1)*	1	1	0
JTAG-based configuration *(3)*	*(2)*	*(2)*	*(2)*

Notes to Table 11–2:

(1) These schemes require that you drive a secondary pin RUnLU to specify whether to perform a remote update or local update.

(2) Do not leave MSEL pins floating. Connect them to V_{CCIO} or GND. These pins support the non-JTAG configuration scheme used in production. If only JTAG configuration is used you should connect the MSEL pins to ground.

(3) JTAG-based configuration takes precedence over other configuration schemes, which means the MSEL pins are ignored.

The MSEL[] pins can be tied to V_{CCIO} of the I/O bank they reside in or ground.

V_{CCSEL} Pins

You can configure Stratix and Stratix GX devices using the 3.3-, 2.5-, 1.8-, or 1.5-V LVTTL I/O standard on configuration and JTAG input pins. VCCSEL is a dedicated input on Stratix and Stratix GX devices that selects between 3.3-V/2.5-V input buffers and 1.8-V/1.5-V input buffers for dedicated configuration input pins. A logic low supports 3.3-V/2.5-V signaling, and a logic high supports 1.8-V/1.5-V signaling. A logic high can also support 3.3-V/2.5-V signaling. VCCSEL affects the configuration

related I/O banks (3, 4, 7, and 8) where the following pins reside: TDI, TMS, TCK, TRST, MSEL0, MSEL1, MSEL2, nCONFIG, nCE, DCLK, PLL_ENA, CONF_DONE, nSTATUS. The VCCSEL pin can be pulled to 1.5, 1.8, 2.5, or 3.3-V for a logic high level. There is an internal 2.5-kΩ pull-down resistor on VCCSEL. Therefore, if you are using a pull-up resister to pull up this signal, you need to use a 1-kΩ resistor.

VCCSEL also sets the power-on-reset (POR) trip point for all the configuration related I/O banks (3, 4, 7, and 8), ensuring that these I/O banks have powered up to the appropriate voltage levels before configuration begins. Upon power-up, the FPGA does not release nSTATUS until V_{CCINT} and all of the V_{CCIO}s of the configuration I/O banks are above their POR trip points. If you set VCCSEL to ground (logic low), this sets the POR trip point for all configuration I/O banks to a voltage consistent with 3.3-V/2.5-V signaling. When VCCSEL = 0, the POR trip point for these I/O banks may be as high as 1.8 V. If V_{CCIO} of any of the configuration banks is set to 1.8 or 1.5 V, the voltage supplied to this I/O bank(s) may never reach the POR trip point, which will not allow the FPGA to begin configuration.

If the V_{CCIO} of I/O banks 3, 4, 7, or 8 is set to 1.5 or 1.8 V and the configuration signals used require 3.3-V or 2.5-V signaling you should set VCCSEL to V_{CC} (logic high) in order to lower the POR trip point to enable successful configuration.

Table 11–3 shows how you should set the VCCSEL depending on the V_{CCIO} setting of the configuration I/O banks and your configuration input signaling voltages.

Table 11–3. VCCSEL Setting

V_{CCIO} (banks 3,4,7,8)	Configuration Input Signaling Voltage	V_{CCSEL}
3.3-V/2.5-V	3.3-V/2.5-V	GND
1.8-V/1.5-V	3.3-V/2.5-V/1.8-V/1.5-V	VCC
3.3-V/2.5-V	1.8-V/1.5-V	Not Supported

The VCCSEL signal does not control any of the dual-purpose pins, including the dual-purpose configuration pins, such as the DATA[7..0] and PPA pins (nWS, nRS, CS, nCS, and RDYnBSY). During configuration, these dual-purpose pins drive out voltage levels corresponding to the V_{CCIO} supply voltage that powers the I/O bank containing the pin. After configuration, the dual-purpose pins inherit the I/O standards specified in the design.

PORSEL Pins

PORSEL is a dedicated input pin used to select POR delay times of 2 ms or 100 ms during power-up. When the PORSEL pin is connected to ground, the POR time is 100 ms; when the PORSEL pin is connected to VCC, the POR time is 2 ms. There is an internal 2.5-kΩ pull-down resistor on PORSEL. Therefore if you are using a pull-up resistor to pull up this signal, you need to use a 1-kΩ resistor.

When using enhanced configuration devices to configure Stratix devices, make sure that the PORSEL setting of the Stratix device is the same or faster than the PORSEL setting of the enhanced configuration device. If the FPGA is not powered up after the enhanced configuration device exits POR, the CONF_DONE signal will be high since the pull-up resistor is pulling this signal high. When the enhanced configuration device exits POR, OE of the enhanced configuration device is released and pulled high by a pull-up resistor. Since the enhanced configuration device sees its nCS/CONF_DONE signal also high, it enters a test mode. Therefore, you must ensure the FPGA powers up before the enhanced configuration device exits POR.

For more margin, the 100-ms setting can be selected when using an enhanced configuration device to allow the Stratix FPGA to power-up before configuration is attempted (see Table 11–4).

Table 11–4. PORSEL Settings

PORSEL Settings	POR Time (ms)
GND	100
V_{CC}	2

nIO_PULLUP Pins

The nIO_PULLUP pin enables a built-in weak pull-up resistor to pull all user I/O pins to VCCIO before and during device configuration. If nIO_PULLUP is connected to V_{CC} during configuration, the weak pull-ups on all user I/O pins and all dual-purpose pins are disabled. If connected to ground, the pull-ups are enabled during configuration. The nIO_PULLUP pin can be pulled to 1.5, 1.8, 2.5, or 3.3-V for a logic level high. There is an internal 2.5-kΩ pull-down resistor on nIO_PULLUP. Therefore, if you are using a pull-up resistor to pull up this signal, you need to use a 1-kΩ resistor.

TDO & nCEO Pins

TDO and nCEO pins drive out the same voltage levels as the V_{CCIO} that powers the I/O bank where the pin resides. You must select the V_{CCIO} supply for the bank containing TDO accordingly. For example, when using the ByteBlasterMV cable, the V_{CCIO} for the bank containing TDO must be powered up at 3.3-V. The current strength for TDO is 12 mA.

Configuration File Size

Tables 11–5 and 11–6 summarize the approximate configuration file size required for each Stratix and Stratix GX device. To calculate the amount of storage space required for multi-device configurations, add the file size of each device together.

Table 11–5. Stratix Configuration File Sizes

Device	Raw Binary File (.rbf) Size (Bits)
EP1S10	3,534,640
EP1S20	5,904,832
EP1S25	7,894,144
EP1S30	10,379,368
EP1S40	12,389,632
EP1S60	17,543,968
EP1S80	23,834,032

Table 11–6. Stratix GX Configuration File Sizes

Device	Raw Binary File Size (Bits)
EP1SGX10C	3,579,928
EP1SGX10D	3,579,928
EP1SGX25C	7,951,248
EP1SGX25D	7,951,248
EP1SGX25F	7,951,248
EP1SGX40D	12,531,440
EP1SGX40G	12,531,440

You should only use the numbers in Tables 11–5 and 11–6 to estimate the file size before design compilation. The exact file size may vary because different Altera® Quartus® II software versions may add a slightly

different number of padding bits during programming. However, for any specific version of the Quartus II software, any design targeted for the same device has the same configuration file size.

Altera Configuration Devices

The Altera enhanced configuration devices (EPC16, EPC8, and EPC4 devices) support a single-device configuration solution for high-density FPGAs and can be used in the FPP and PS configuration schemes. They are ISP-capable through its JTAG interface. The enhanced configuration devices are divided into two major blocks, the controller and the flash memory.

For information on enhanced configuration devices, see the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* and the *Using Altera Enhanced Configuration Devices* chapter in the *Configuration Handbook*.

The EPC2 and EPC1 configuration devices provide configuration support for the PS configuration scheme. The EPC2 device is ISP-capable through its JTAG interface. The EPC2 and EPC1 can be cascaded to hold large configuration files.

For more information on EPC2, EPC1, and EPC1441 configuration devices, see the *Configuration Devices for SRAM-Based LUT Devices Data Sheet*.

Configuration Schemes

This section describes how to configure Stratix and Stratix GX devices with the following configuration schemes:

- PS Configuration with Configuration Devices
- PS Configuration with a Download Cable
- PS Configuration with a Microprocessor
- FPP Configuration
- PPA Configuration
- JTAG Programming & Configuration
- JTAG Programming & Configuration of Multiple Devices

PS Configuration

PS configuration of Stratix and Stratix GX devices can be performed using an intelligent host, such as a MAX® device, microprocessor with flash memory, an Altera configuration device, or a download cable. In the PS scheme, an external host (MAX device, embedded processor, configuration device, or host PC) controls configuration. Configuration data is clocked into the target Stratix devices via the `DATA0` pin at each rising edge of `DCLK`.

PS Configuration with Configuration Devices

The configuration device scheme uses an Altera configuration device to supply data to the Stratix or Stratix GX device in a serial bitstream (see Figure 11–3).

In the configuration device scheme, nCONFIG is usually tied to V_{CC} (when using EPC16, EPC8, EPC4, or EPC2 devices, nCONFIG may be connected to nINIT_CONF). Upon device power-up, the target Stratix or Stratix GX device senses the low-to-high transition on nCONFIG and initiates configuration. The target device then drives the open-drain CONF_DONE pin low, which in-turn drives the configuration device's nCS pin low. When exiting power-on reset (POR), both the target and configuration device release the open-drain nSTATUS pin.

Before configuration begins, the configuration device goes through a POR delay of up to 200 ms to allow the power supply to stabilize (power the Stratix or Stratix GX device before or during the POR time of the configuration device). This POR delay has a maximum of 200 ms for EPC2 devices. For enhanced configuration devices, you can select between 2 ms and 100 ms by connecting PORSEL pin to VCC or GND, accordingly. During this time, the configuration device drives its OE pin low. This low signal delays configuration because the OE pin is connected to the target device's nSTATUS pin. When the target and configuration devices complete POR, they release nSTATUS, which is then pulled high by a pull-up resistor.

When configuring multiple devices, configuration does not begin until all devices release their OE or nSTATUS pins. When all devices are ready, the configuration device clocks data out serially to the target devices using an internal oscillator.

After successful configuration, the Stratix FPGA starts initialization using the 10-MHz internal oscillator as the reference clock. After initialization, this internal oscillator is turned off. The CONF_DONE pin is released by the target device and then pulled high by a pull-up resistor. When initialization is complete, the FPGA enters user mode. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

If an error occurs during configuration, the target device drives its nSTATUS pin low, resetting itself internally and resetting the configuration device. If the **Auto-Restart Configuration on Frame Error** option—available in the Quartus II **Global Device Options** dialog box (Assign menu)—is turned on, the device reconfigures automatically if an error occurs. To find this option, choose **Compiler Settings** (Processing menu), then click on the **Chips & Devices** tab.

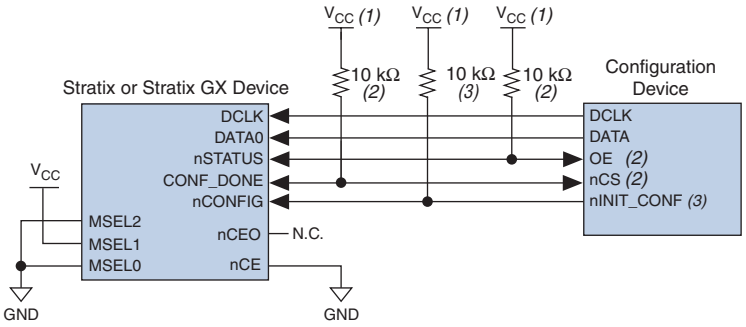
If this option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low to restart configuration. The external system can pulse nCONFIG if it is under system control rather than tied to V_{CC}. When configuration is complete, the target device releases CONF_DONE, which disables the configuration device by driving nCS high. The configuration device drives DCLK low before and after configuration.

In addition, if the configuration device sends all of its data and then detects that CONF_DONE has not gone high, it recognizes that the target device has not configured successfully. In this case, the configuration device pulses its OE pin low for a few microseconds, driving the target device's nSTATUS pin low. If the **Auto-Restart Configuration on Frame Error** option is set in the software, the target device resets and then pulses its nSTATUS pin low. When nSTATUS returns high, the configuration device reconfigures the target device. When configuration is complete, the configuration device drives DCLK low.

Do not pull CONF_DONE low to delay initialization. Instead, use the Quartus II software's **Enable User-Supplied Start-Up Clock (CLKUSR)** option to synchronize the initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain initialize together. When CONF_DONE is driven low after device configuration, the configuration device recognizes that the target device has not configured successfully.

Figure 11–2 shows how to configure one Stratix or Stratix GX device with one configuration device.

Figure 11–2. Single Device Configuration Circuit



Notes to Figure 11–2:

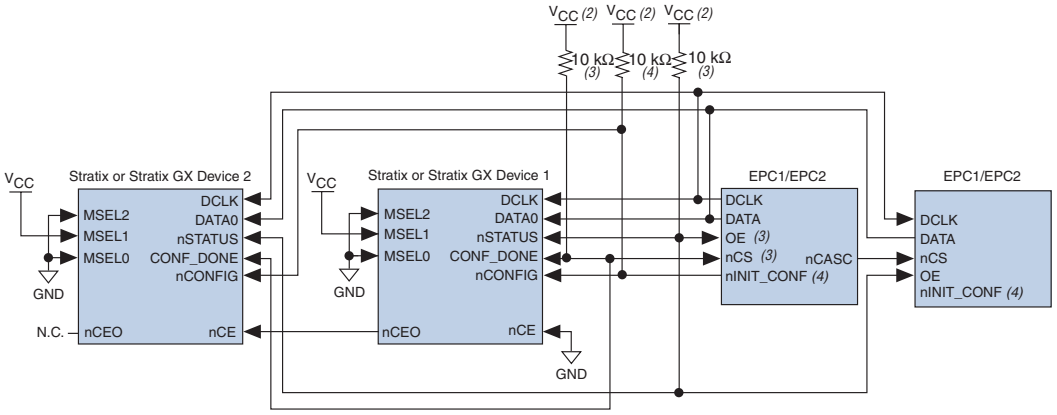
(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The enhanced configuration devices and EPC2 devices have internal programmable pull-ups on OE and nCS. You should only use the internal pull-ups of the configuration device if the nSTATUS and CONF_DONE signals are pulled up to 3.3 V or 2.5 V (not 1.8 V or 1.5 V). If external pull-ups are used, they should be 10 kΩ

(3) The nINIT_CONF pin is available on EPC16, EPC8, EPC4, and EPC2 devices. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} through a resistor. he nINIT_CONF pin has an internal pull-up resistor that is always active in EPC16, EPC8, EPC4, and EPC2 devices. These devices do not need an external pull-up resistor on the nINIT_CONF pin.

Figure 11–3 shows how to configure multiple Stratix and Stratix GX devices with multiple EPC2 or EPC1 configuration devices.

Figure 11–3. Multi-Device Configuration Circuit Note (1)



Notes to Figure 11–3:

(1) When performing multi-device active serial configuration, you must generate the configuration device programmer object file (**.pof**) from each project's SOF. You can combine multiple SOFs using the Quartus II software through the **Device & Pin Option** dialog box. For more information on how to create configuration and programming files, see the *Software Settings* section in the *Configuration Handbook, Volume 2*.

(2) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(3) The enhanced configuration devices and EPC2 devices have internal programmable pull-ups on OE and nCS. You should only use the internal pull-ups of the configuration device if the nSTATUS and CONF_DONE signals are pulled up to 3.3 V or 2.5 V (not 1.8 V or 1.5 V). If external pull-ups are used, they should be 10 kΩ

(4) The nINIT_CONF pin is available on EPC16, EPC8, EPC4, and EPC2 devices. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} through a resistor. The nINIT_CONF pin has an internal pull-up resistor that is always active in EPC16, EPC8, EPC4, and EPC2 devices. These devices do not need an external pull-up resistor on the nINIT_CONF pin.

After the first Stratix or Stratix GX device completes configuration during multi-device configuration, its nCEO pin activates the second device's nCE pin, prompting the second device to begin configuration. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

In addition, all nSTATUS pins are tied together; thus, if any device (including the configuration devices) detects an error, configuration stops for the entire chain. Also, if the first configuration device does not detect CONF_DONE going high at the end of configuration, it resets the chain by pulsing its OE pin low for a few microseconds. This low pulse drives the OE pin low on the second configuration device and drives nSTATUS low on all Stratix and Stratix GX devices, causing them to enter an error state.

If the **Auto-Restart Configuration on Frame Error** option is turned on in the software, the Stratix or Stratix GX device releases its nSTATUS pins after a reset time-out period. When the nSTATUS pins are released and pulled high, the configuration devices reconfigure the chain. If the **Auto-**

Restart Configuration on Frame Error option is not turned on, the Stratix or Stratix GX devices drive nSTATUS low until they are reset with a low pulse on nCONFIG.

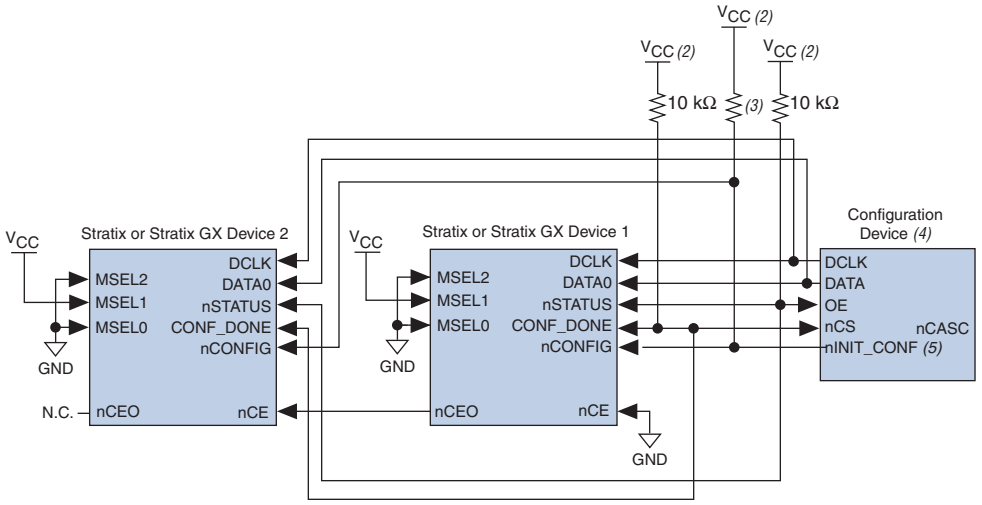
You can also cascade several EPC2/EPC1 configuration devices to configure multiple Stratix and Stratix GX devices. When all data from the first configuration device is sent, it drives nCASC low, which in turn drives nCS on the subsequent configuration device. Because a configuration device requires less than one clock cycle to activate a subsequent configuration device, the data stream is uninterrupted.

You cannot cascade enhanced (EPC16, EPC8, and EPC4) configuration devices.

You can use a single configuration chain to configure multiple Stratix and Stratix GX devices. In this scheme, the nCEO pin of the first device is connected to the nCE pin of the second device in the chain. If there are additional devices, connect the nCE pin of the next device to the nCEO pin of the previous device. To configure properly, all of the device CONF_DONE and nSTATUS pins must be tied together.

Figure 11–4 shows an example of configuring multiple Stratix and Stratix GX devices using a configuration device.

Figure 11–4. Configuring Multiple Stratix & Stratix GX Devices with A Single Configuration Device Note (1)



Notes to Figure 11–4:

(1) When performing multi-device active serial configuration, you must generate the configuration device programmer object file (**.pof)** from each project's SOF. You can combine multiple SOFs using the Quartus II software through the **Device & Pin Option** dialog box. For more information on how to create configuration and programming files, see the *Software Settings* section in the *Configuration Handbook, Volume 2*.

(2) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(3) The enhanced configuration devices and EPC2 devices have internal programmable pull-ups on `OE` and `nCS`. You should only use the internal pull-ups of the configuration device if the `nSTATUS` and `CONF_DONE` signals are pulled up to 3.3 V or 2.5 V (not 1.8 V or 1.5 V). If external pull-ups are used, they should be 10 kΩ

(4) EPC16, EPC8, and EPC4 configuration devices cannot be cascaded.

(5) The `nINIT_CONF` pin is available on EPC16, EPC8, EPC4, and EPC2 devices. If `nINIT_CONF` is not used, `nCONFIG` must be pulled to V_{CC} through a resistor. The `nINIT_CONF` pin has an internal pull-up resistor that is always active in EPC16, EPC8, EPC4, and EPC2 devices. These devices do not need an external pull-up resistor on the `nINIT_CONF` pin.

Table 11–7 shows the status of the device DATA pins during and after configuration.

Table 11–7. DATA Pin Status Before & After Configuration

Pins	Stratix or Stratix GX Device	
	During	After
DATA0 *(1)*	Used for configuration	User defined
DATA[7..1] *(2)*	Used in some configuration modes	User defined
I/O Pins	Tri-state	User defined

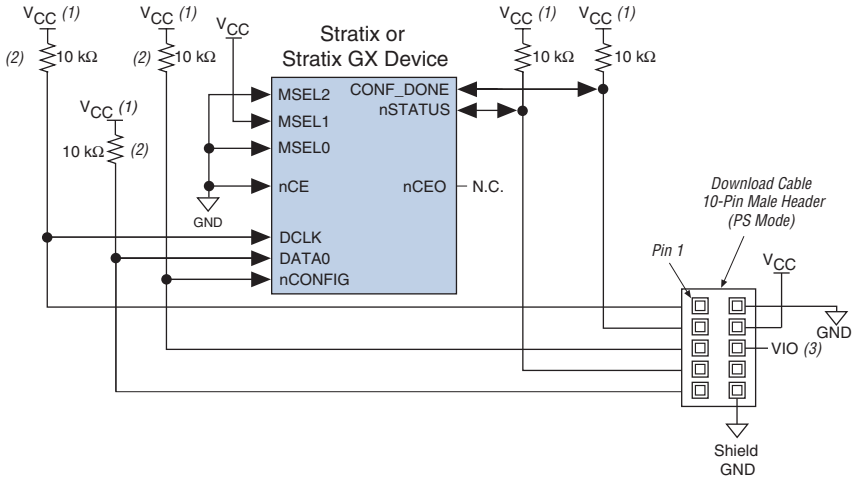
Notes to Table 11–7:
(1) The status shown is for configuration with a configuration device.
(2) The function of these pins depends upon the settings specified in the Quartus II software using the **Device & Pin Option** dialog box (see the *Software Settings* section in the *Configuration Handbook, Volume 2*, and the Quartus II Help software for more information).

PS Configuration with a Download Cable

In PS configuration with a download cable, an intelligent host transfers data from a storage device to the Stratix or Stratix GX device through the MasterBlaster, USB-Blaster, ByteBlaster II or ByteBlasterMV cable. To initiate configuration in this scheme, the download cable generates a low-to-high transition on the nCONFIG pin. The programming hardware then places the configuration data one bit at a time on the device's DATA0 pin. The data is clocked into the target device until CONF_DONE goes high. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

When using programming hardware for the Stratix or Stratix GX device, turning on the **Auto-Restart Configuration on Frame Error** option does not affect the configuration cycle because the Quartus II software must restart configuration when an error occurs. Additionally, the **Enable User-Supplied Start-Up Clock (CLKUSR)** option has no affect on the device initialization since this option is disabled in the SOF when programming the FPGA using the Quartus II software programmer and a download cable. Therefore, if you turn on the **CLKUSR** option, you do not need to provide a clock on CLKUSR when you are configuring the FPGA with the Quartus II programmer and a download cable.
Figure 11–5 shows PS configuration for the Stratix or Stratix GX device using a MasterBlaster, USB-Blaster, ByteBLaster II or ByteBlasterMV cable.

Figure 11–5. PS Configuration Circuit with a Download Cable



Notes to Figure 11–5:

(1) You should connect the pull-up resistor to the same supply voltage as the MasterBlaster (VIO pin) or ByteBlasterMV cable.

(2) The pull-up resistors on the DATA0 and DCLK pins are only needed if the download cable is the only configuration scheme used on the board. This is to ensure that the DATA0 and DCLK pins are not left floating after configuration. For example, if the design also uses a configuration device, the pull-up resistors on the DATA0 and DCLK pins are not necessary.

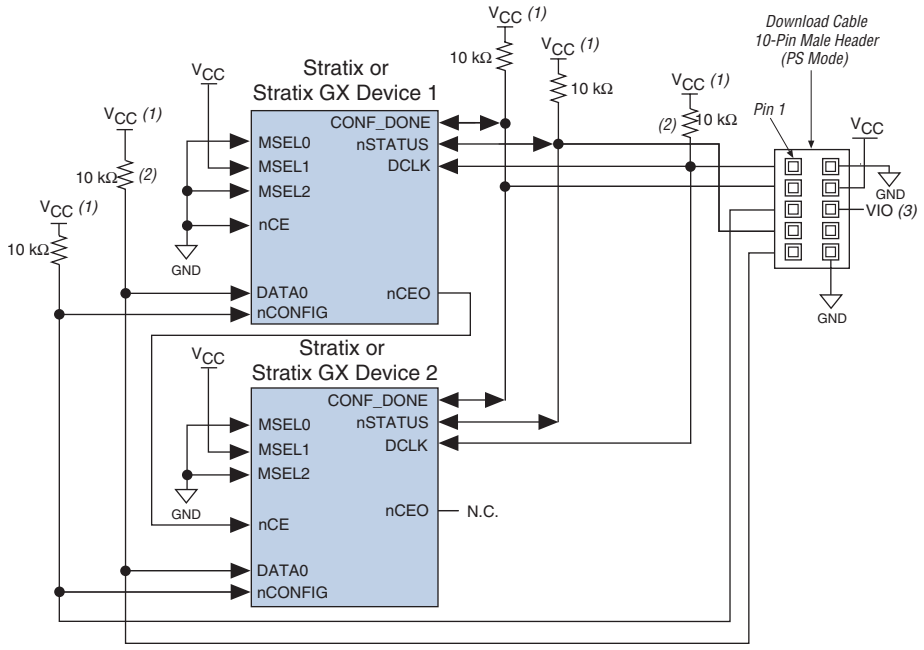
(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. This pin is a no-connect pin for the ByteBlasterMV header.

You can use programming hardware to configure multiple Stratix and Stratix GX devices by connecting each device's nCEO pin to the subsequent device's nCE pin. All other configuration pins are connected to each device in the chain.

Because all CONF_DONE pins are tied together, all devices in the chain initialize and enter user mode at the same time. In addition, because the nSTATUS pins are tied together, the entire chain halts configuration if any device detects an error. In this situation, the Quartus II software must restart configuration; the **Auto-Restart Configuration on Frame Error** option does not affect the configuration cycle.

Figure 11–6 shows how to configure multiple Stratix and Stratix GX devices with a MasterBlaster or ByteBlasterMV cable.

Figure 11–6. Multi-Device PS Configuration with a Download Cable



Notes to Figure 11–6:

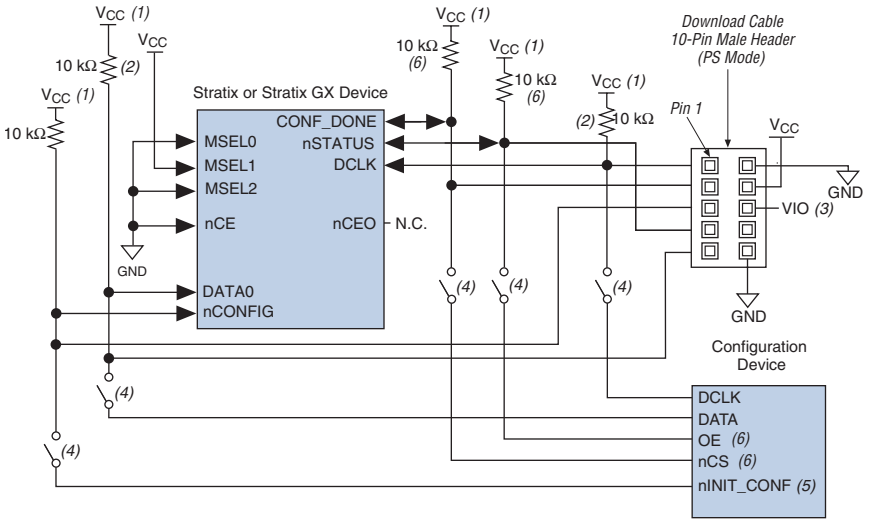
(1) You should connect the pull-up resistor to the same supply voltage as the MasterBlaster (VIO pin) or ByteBlasterMV cable.

(2) The pull-up resistors on the DATA0 and DCLK pins are only needed if the download cable is the only configuration scheme used on the board. This is to ensure that the DATA0 and DCLK pins are not left floating after configuration. For example, if the design also uses a configuration device, the pull-up resistors on the DATA0 and DCLK pins are not necessary.

(3) V_{IO} is a reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. See the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value.

If you are using a download cable to configure device(s) on a board that also has configuration devices, you should electrically isolate the configuration devices from the target device(s) and cable. One way to isolate the configuration devices is to add logic, such as a multiplexer, that can select between the configuration devices and the cable. The multiplexer device should allow bidirectional transfers on the nSTATUS and CONF_DONE signals. Another option is to add switches to the five common signals (CONF_DONE, nSTATUS, DCLK, nCONFIG, and DATA0) between the cable and the configuration devices. The last option is to remove the configuration devices from the board when configuring with the cable. Figure 11–7 shows a combination of a configuration device and a download cable to configure a Stratix or Stratix GX device.

Figure 11–7. Configuring with a Combined PS & Configuration Device Scheme



Notes to Figure 11–7:

(1) You should connect the pull-up resistor to the same supply voltage as the configuration device.

(2) The pull-up resistors on the DATA0 and DCLK pins are only needed if the download cable is the only configuration scheme used on the board. This is to ensure that the DATA0 and DCLK pins are not left floating after configuration. For example, if the design also uses a configuration device, the pull-up resistors on the DATA0 and DCLK pins are not necessary.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the target device's V_{CCIO}. This is a no-connect pin for the ByteBlasterMV header.

(4) You should not attempt configuration with a download cable while a configuration device is connected to a Stratix or Stratix GX device. Instead, you should either remove the configuration device from its socket when using the download cable or place a switch on the five common signals between the download cable and the configuration device. Remove the download cable when configuring with a configuration device.

(5) If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(6) If external pull-ups are used on CONF_DONE and nSTATUS pins, they should always be 10 kΩ resistors. You can use the internal pull-ups of the configuration device only if the CONF_DONE and nSTATUS signals are pulled-up to 3.3 V or 2.5 V (not 1.8 V or 1.5 V).

For more information on how to use the MasterBlaster or ByteBlasterMV cables, see the following documents:

- *USB-Blaster USB Port Download Cable Data Sheet*
- *MasterBlaster Serial/USB Communications Cable Data Sheet*
- *ByteBlasterMV Parallel Port Download Cable Data Sheet*
- *ByteBlaster II Parallel Port Download Cable Data Sheet*

PS Configuration with a Microprocessor

In PS configuration with a microprocessor, a microprocessor transfers data from a storage device to the target Stratix or Stratix GX device. To initiate configuration in this scheme, the microprocessor must generate a low-to-high transition on the nCONFIG pin and the target device must release nSTATUS. The microprocessor or programming hardware then places the configuration data one bit at a time on the DATA0 pin of the Stratix or Stratix GX device. The least significant bit (LSB) of each data byte must be presented first. Data is clocked continuously into the target device until CONF_DONE goes high.

After all configuration data is sent to the Stratix or Stratix GX device, the CONF_DONE pin goes high to show successful configuration and the start of initialization. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize. Initialization, by default, uses an internal oscillator, which runs at 10 MHz. After initialization, this internal oscillator is turned off. If you are using the **clkusr** option, after all data is transferred clkusr must be clocked an additional 136 times for the Stratix or Stratix GX device to initialize properly. Driving DCLK to the device after configuration is complete does not affect device operation.

Handshaking signals are not used in PS configuration modes. Therefore, the configuration clock speed must be below the specified frequency to ensure correct configuration. No maximum DCLK period exists. You can pause configuration by halting DCLK for an indefinite amount of time.

If the target device detects an error during configuration, it drives its nSTATUS pin low to alert the microprocessor. The microprocessor can then pulse nCONFIG low to restart the configuration process. Alternatively, if the **Auto-Restart Configuration on Frame Error** option is turned on in the Quartus II software, the target device releases nSTATUS after a reset time-out period. After nSTATUS is released, the microprocessor can reconfigure the target device without needing to pulse nCONFIG low.

The microprocessor can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. If the microprocessor sends all data and the initialization clock starts but CONF_DONE and INIT_DONE have not gone high, it must reconfigure the target device. By default the INIT_DONE output is disabled. You can enable the INIT_DONE output by turning on **Enable INIT_DONE output** option in the Quartus II software.

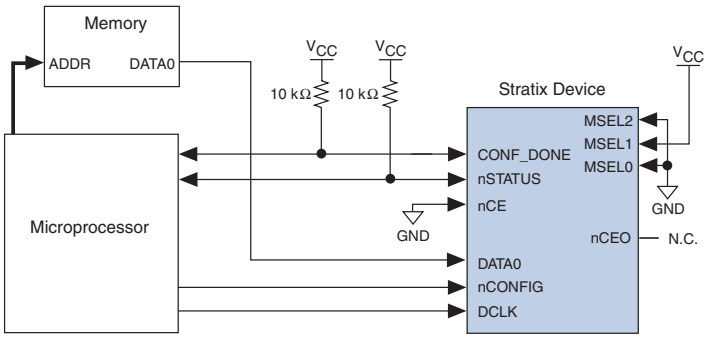
If you do not turn on the **Enable INIT_DONE output** option in the Quartus II software, you are advised to wait for the maximum value of t_{CD2UM} (see Table 11–8) after the CONF_DONE signal goes high to ensure the device has been initialized properly and that it has entered user mode.

During configuration and initialization, and before the device enters user mode, the microprocessor must not drive the CONF_DONE signal low.

If the optional CLKUSR pin is used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure CLKUSR continues toggling during the time nSTATUS is low (maximum of 40 µs).

Figure 11–8 shows the circuit for PS configuration with a microprocessor.

Figure 11–8. PS Configuration Circuit with Microprocessor



PS Configuration Timing

Figure 11–9 shows the PS configuration timing waveform for Stratix and Stratix GX devices. Table 11–8 shows the PS timing parameters for Stratix and Stratix GX devices.

Table 11–8. PS Timing Parameters for Stratix & Stratix GX Devices

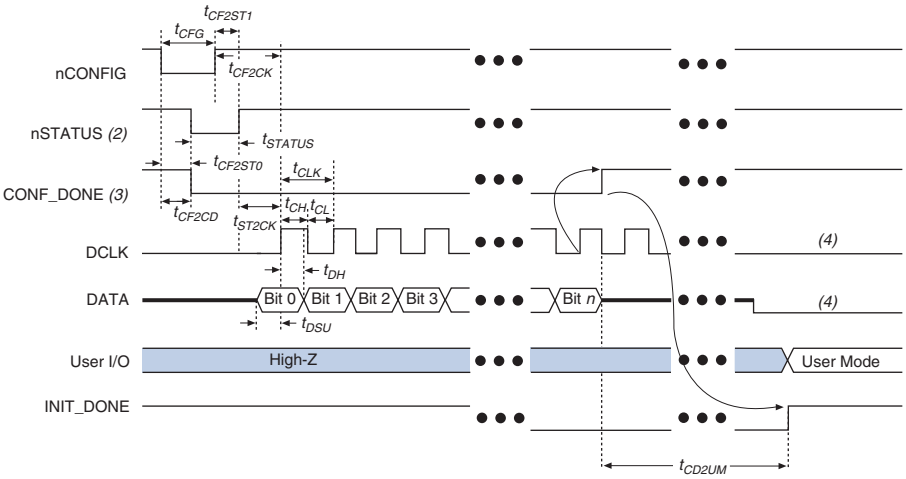
Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		800	ns
t_{CF2ST1}	nCONFIG high to nSTATUS high		40 *(2)*	µs
t_{CFG}	nCONFIG low pulse width	40		µs
t_{STATUS}	nSTATUS low pulse width	10	40 *(2)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	40		µs
t_{ST2CK}	nSTATUS high to first rising edge on DCLK	1		µs
t_{DSU}	Data setup time before rising edge on DCLK	7		ns
t_{DH}	Data hold time after rising edge on DCLK	0		ns
t_{CH}	DCLK high time	4		ns
t_{CL}	DCLK low time	4		ns
t_{CLK}	DCLK period	10		ns
f_{MAX}	DCLK maximum frequency		100	MHz
t_{CD2UM}	CONF_DONE high to user mode *(1)*	6	20	µs

Notes to Table 11–8:

(1) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device. If the clock source is CLKUSR, multiply the clock period by 136 to obtain this value.

(2) This value is obtainable if users do not delay configuration by extending the nSTATUS low pulse width.

Figure 11–9. PS Timing Waveform for Stratix & Stratix GX Devices Note (1)



Notes to Figure 11–9:

(1) The beginning of this waveform shows the device in user-mode. In user-mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.
(2) Upon power-up, the Stratix II device holds nSTATUS low for the time of the POR delay.
(3) Upon power-up, before and during configuration, CONF_DONE is low.
(4) DCLK should not be left floating after configuration. It should be driven high or low, whichever is convenient. DATA[] is available as user I/Os after configuration and the state of these pins depends on the dual-purpose pin settings.

FPP Configuration

Parallel configuration of Stratix and Stratix GX devices meets the continuously increasing demand for faster configuration times. Stratix and Stratix GX devices can receive byte-wide configuration data per clock cycle, and guarantee a configuration time of less than 100 ms with a 100-MHz configuration clock. Stratix and Stratix GX devices support programming data bandwidth up to 800 megabits per second (Mbps) in this mode. You can use parallel configuration with an EPC16, EPC8, or EPC4 device, or a microprocessor.

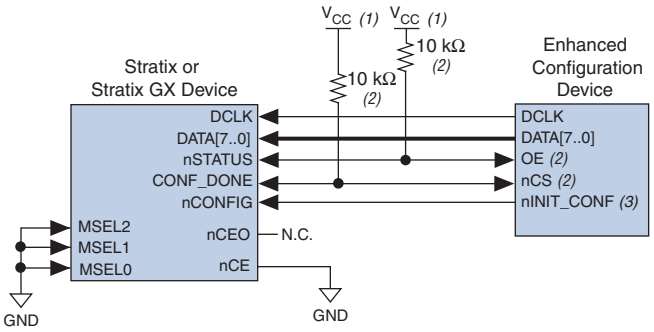
This section discusses the following schemes for FPP configuration in Stratix and Stratix GX devices:

- FPP Configuration Using an Enhanced Configuration Device
- FPP Configuration Using a Microprocessor

FPP Configuration Using an Enhanced Configuration Device

When using FPP with an enhanced configuration device, it supplies data in a byte-wide fashion to the Stratix or Stratix GX device every DCLK cycle. See Figure 11–10.

Figure 11–10. FPP Configuration Using Enhanced Configuration Devices



Notes to Figure 11–10:

(1) The pull-up resistors should be connected to the same supply voltage as the configuration device.

(2) The enhanced configuration devices and EPC2 devices have internal programmable pull-ups on OE and nCS. You should only use the internal pull-ups of the configuration device if the nSTATUS and CONF_DONE signals are pulled up to 3.3 V or 2.5 V (not 1.8 V or 1.5 V). If external pull-ups are used, they should be 10 kΩ

(3) The nINIT_CONF pin is available on EPC16, EPC8, EPC4, and EPC2 devices. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} through a resistor. The nINIT_CONF pin has an internal pull-up resistor that is always active in EPC16, EPC8, EPC4, and EPC2 devices. These devices do not need an external pull-up resistor on the nINIT_CONF pin.

In the enhanced configuration device scheme, nCONFIG is tied to nINIT_CONF. On power up, the target Stratix or Stratix GX device senses the low-to-high transition on nCONFIG and initiates configuration. The target Stratix or Stratix GX device then drives the open-drain CONF_DONE pin low, which in-turn drives the enhanced configuration device's nCS pin low.

Before configuration starts, there is a 2-ms POR delay if the PORSEL pin is connected to V_{CC} in the enhanced configuration device. If the PORSEL pin is connected to ground, the POR delay is 100 ms. When each device determines that its power is stable, it releases its nSTATUS or OE pin. Because the enhanced configuration device's OE pin is connected to the target Stratix or Stratix GX device's nSTATUS pin, configuration is delayed until both the nSTATUS and OE pins are released by each device. The nSTATUS and OE pins are pulled up by a resistor on their respective

devices once they are released. When configuring multiple devices, connect the nSTATUS pins together to ensure configuration only happens when all devices release their OE or nSTATUS pins. The enhanced configuration device then clocks data out in parallel to the Stratix or Stratix GX device using a 66-MHz internal oscillator, or drives it to the Stratix or Stratix GX device through the EXTCLK pin.

If there is an error during configuration, the Stratix or Stratix GX device drives the nSTATUS pin low, resetting itself internally and resetting the enhanced configuration device. The Quartus II software provides an **Auto-restart configuration after error** option that automatically initiates the reconfiguration whenever an error occurs. See the *Software Settings* chapter in Volume 2 of the *Configuration Handbook* for information on how to turn this option on or off.

If this option is turned off, you must set monitor nSTATUS to check for errors. To initiate reconfiguration, pulse nCONFIG low. The external system can pulse nCONFIG if it is under system control rather than tied to V_{CC}. Therefore, nCONFIG must be connected to nINIT_CONF if you want to reprogram the Stratix or Stratix GX device on the fly.

When configuration is complete, the Stratix or Stratix GX device releases the CONF_DONE pin, which is then pulled up by a resistor. This action disables the EPC16, EPC8, or EPC4 enhanced configuration device as nCS is driven high. Initialization, by default, uses an internal oscillator, which runs at 10 MHz. After initialization, this internal oscillator is turned off. When initialization is complete, the Stratix or Stratix GX device enters user mode. The enhanced configuration device drives DCLK low before and after configuration.

CONF_DONE goes high one byte early in parallel synchronous (FPP) and asynchronous (PPA) modes using a microprocessor with **.rbf**, **.hex**, and **.ttf** file formats. This does not apply to FPP mode for enhanced configuration devices using **.pof** file format. This also does not apply to serial modes.

If, after sending out all of its data, the enhanced configuration device does not detect CONF_DONE going high, it recognizes that the Stratix or Stratix GX device has not configured successfully. The enhanced configuration device pulses its OE pin low for a few microseconds, driving the nSTATUS pin on the Stratix or Stratix GX device low. If the **Auto-restart configuration after error** option is on, the Stratix or Stratix GX device resets and then pulses its nSTATUS low. When nSTATUS returns high, reconfiguration is restarted (see Figure 11–11 on page 11–25).

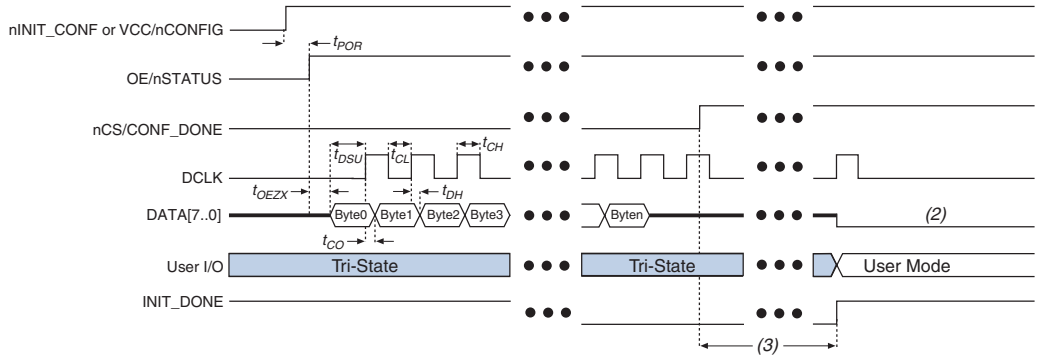
Do not drive CONF_DONE low after device configuration to delay initialization. Instead, use the **Enable User-Supplied Start-Up Clock (CLKUSR)** option in the **Device & Pin Options** dialog box. You can use this option to synchronize the initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain initialize together.

After the first Stratix or Stratix GX device completes configuration during multi-device configuration, its nCEO pin activates the second Stratix or Stratix GX device's nCE pin, prompting the second device to begin configuration. Because CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time. Because nSTATUS pins are tied together, configuration stops for the whole chain if any device (including enhanced configuration devices) detects an error. Also, if the enhanced configuration device does not detect a high on CONF_DONE at the end of configuration, it pulses its OE low for a few microseconds to reset the chain. The low OE pulse drives nSTATUS low on all Stratix and Stratix GX devices, causing them to enter an error state. This state is similar to a Stratix or Stratix GX device detecting an error.

If the **Auto-restart configuration after error** option is on, the Stratix and Stratix GX devices release their nSTATUS pins after a reset time-out period. When the nSTATUS pins are released and pulled high, the configuration device reconfigures the chain. If the **Auto-restart configuration after error** option is off, nSTATUS stays low until the Stratix and Stratix GX devices are reset with a low pulse on nCONFIG.

Figure 11–11 shows the FPP configuration with a configuration device timing waveform for Stratix and Stratix GX devices.

Figure 11–11. FPP Configuration with a Configuration Device Timing Waveform Note (1)



Notes to Figure 11–11:

(1) For timing information, see the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet*.
(2) The configuration device drives DATA high after configuration.
(3) Stratix and Stratix GX devices enter user mode 136 clock cycles after CONF_DONE goes high.

FPP Configuration Using a Microprocessor

When using a microprocessor for parallel configuration, the microprocessor transfers data from a storage device to the Stratix or Stratix GX device through configuration hardware. To initiate configuration, the microprocessor needs to generate a low-to-high transition on the nCONFIG pin and the Stratix or Stratix GX device must release nSTATUS. The microprocessor then places the configuration data to the DATA[7..0] pins of the Stratix or Stratix GX device. Data is clocked continuously into the Stratix or Stratix GX device until CONF_DONE goes high.

The configuration clock (DCLK) speed must be below the specified frequency to ensure correct configuration. No maximum DCLK period exists. You can pause configuration by halting DCLK for an indefinite amount of time.

After all configuration data is sent to the Stratix or Stratix GX device, the CONF_DONE pin goes high to show successful configuration and the start of initialization. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize. Initialization, by default, uses an internal oscillator, which runs at 10 MHz. After initialization, this internal oscillator is turned off. If you are using the **clkusr** option, after all data is transferred clkusr must be clocked an additional 136 times for the Stratix or Stratix GX device to initialize properly. Driving DCLK to the device after configuration is complete does not affect device operation. By

default, the INIT_DONE output is disabled. You can enable the INIT_DONE output by turning on the **Enable INIT_DONE output** option in the Quartus II software.

If you do not turn on the **Enable INIT_DONE output** option in the Quartus II software, you are advised to wait for maximum value of t_{CD2UM} (see Table 11–9) after the CONF_DONE signal goes high to ensure the device has been initialized properly and that it has entered user mode.

During configuration and initialization and before the device enters user mode, the microprocessor must not drive the CONF_DONE signal low.

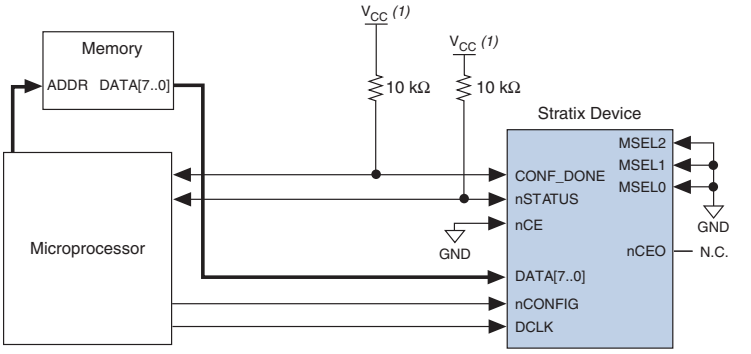
If the optional CLKUSR pin is used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure CLKUSR continues toggling during the time nSTATUS is low (maximum of 40 µs).

If the Stratix or Stratix GX device detects an error during configuration, it drives nSTATUS low to alert the microprocessor. The pin on the microprocessor connected to nSTATUS must be an input. The microprocessor can then pulse nCONFIG low to restart the configuration error. With the **Auto-restart configuration after error** option on, the Stratix or Stratix GX device releases nSTATUS after a reset time-out period. After nSTATUS is released, the microprocessor can reconfigure the Stratix or Stratix GX device without pulsing nCONFIG low.

The microprocessor can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. If the microprocessor sends all the data and the initialization clock starts but CONF_DONE and INIT_DONE have not gone high, it must reconfigure the Stratix or Stratix GX device. After waiting the specified 136 DCLK cycles, the microprocessor should restart configuration by pulsing nCONFIG low.

Figure 11–12 shows the circuit for Stratix and Stratix GX parallel configuration using a microprocessor.

Figure 11–12. Parallel Configuration Using a Microprocessor



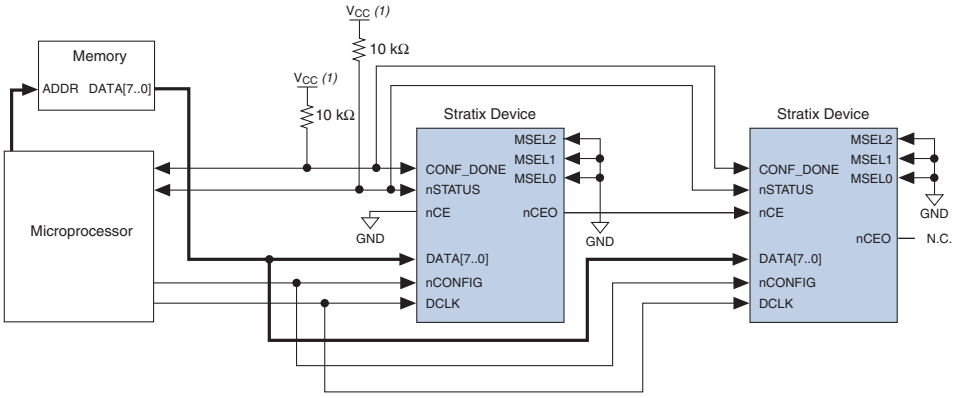
Note to Figure 11–12:

(1) The pull-up resistors should be connected to any V_{CC} that meets the Stratix high-level input voltage (V_{IH}) specification.

For multi-device parallel configuration with a microprocessor, the nCEO pin of the first Stratix or Stratix GX device is cascaded to the second device's nCE pin. The second device in the chain begins configuration within one clock cycle; therefore, the transfer of data destinations is transparent to the microprocessor. Because the CONF_DONE pins of the devices are connected together, all devices initialize and enter user mode at the same time.

Because the nSTATUS pins are also tied together, if any of the devices detects an error, the entire chain halts configuration and drives nSTATUS low. The microprocessor can then pulse nCONFIG low to restart configuration. If the **Auto-restart configuration after error** option is on, the Stratix and Stratix GX devices release nSTATUS after a reset time-out period. The microprocessor can then reconfigure the devices once nSTATUS is released. Figure 11–13 shows multi-device configuration using a microprocessor. Figure 11–14 shows multi-device configuration when both Stratix and Stratix GX devices are receiving the same data. In this case, the microprocessor sends the data to both devices simultaneously, and the devices configure simultaneously.

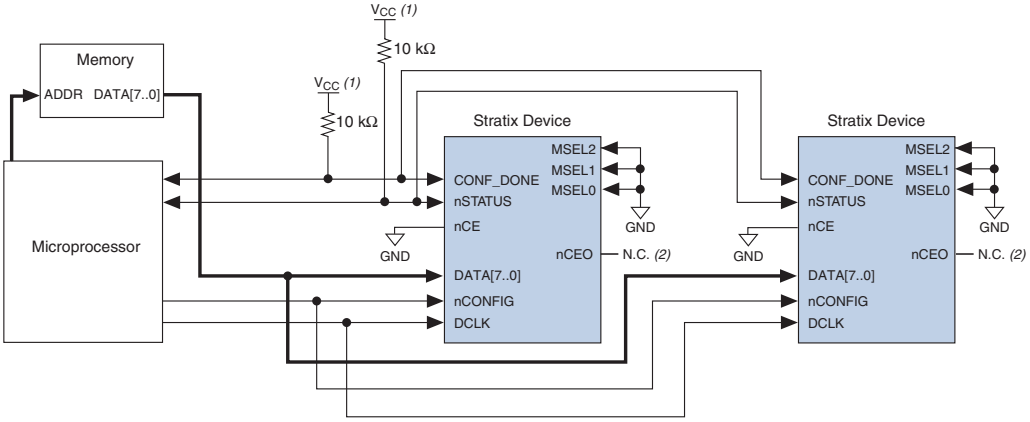
Figure 11–13. Parallel Data Transfer in Serial Configuration with a Microprocessor



Note to Figure 11–13:

(1) You should connect the pull-up resistors to any V_{CC} that meets the Stratix high-level input voltage (V_{IH}) specification.

Figure 11–14. Multiple Device Parallel Configuration with the Same Data Using a Microprocessor



Notes to Figure 11–14:

(1) You should connect the pull-up resistors to any V_{CC} that meets the Stratix high-level input voltage (V_{IH}) specification.

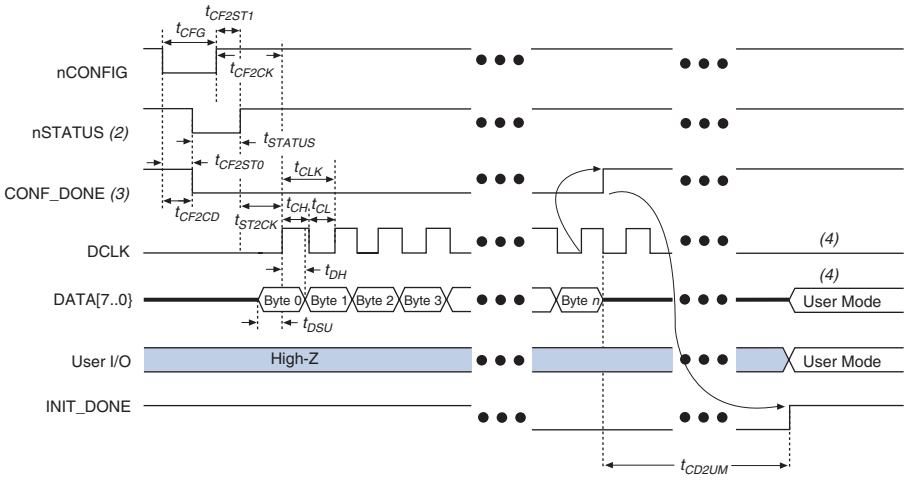
(2) The nCEO pins are left unconnected when configuring the same data into multiple Stratix or Stratix GX devices.

For more information on configuring multiple Altera devices in the same configuration chain, see the *Configuring Mixed Altera FPGA Chains* chapter in the *Configuration Handbook, Volume 2*.

FPP Configuration Timing

Figure 11–15 shows FPP timing waveforms for configuring a Stratix or Stratix GX device in FPP mode. Table 11–9 shows the FPP timing parameters for Stratix or Stratix GX devices.

Figure 11–15. Timing Waveform for Configuring Devices in FPP Mode *Note (1)*



Notes to Figure 11–15:

(1) The beginning of this waveform shows the device in user-mode. In user-mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.
(2) Upon power-up, the Stratix II device holds nSTATUS low for the time of the POR delay.
(3) Upon power-up, before and during configuration, CONF_DONE is low.
(4) DCLK should not be left floating after configuration. It should be driven high or low, whichever is convenient. DATA[] is available as user I/Os after configuration and the state of these pins depends on the dual-purpose pin settings.

Table 11–9. FPP Timing Parameters for Stratix & Stratix GX Devices (Part 1 of 2)

Symbol	Parameter	Min	Max	Units
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	40		µs
t_{DSU}	Data setup time before rising edge on DCLK	7		ns
t_{DH}	Data hold time after rising edge on DCLK	0		ns
t_{CFG}	nCONFIG low pulse width	40		µs
t_{CH}	DCLK high time	4		ns
t_{CL}	DCLK low time	4		ns
t_{CLK}	DCLK period	10		ns

Table 11–9. FPP Timing Parameters for Stratix & Stratix GX Devices (Part 2 of 2)

Symbol	Parameter	Min	Max	Units
f_{MAX}	DCLK frequency		100	MHz
t_{CD2UM}	CONF_DONE high to user mode *(1)*	6	20	µs
t_{CF2CD}	nCONFIG low to CONF_DONE low		800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		800	ns
t_{CF2ST1}	nCONFIG high to nSTATUS high		40 *(2)*	µs
t_{STATUS}	nSTATUS low pulse width	10	40 *(2)*	µs
t_{ST2CK}	nSTATUS high to firstrising edge of DCLK	1		µs

Notes to Table 11–9:

(1) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device. If the clock source is CLKUSR, multiply the clock period by 136 to obtain this value.

(2) This value is obtainable if users do not delay configuration by extending the nSTATUS low pulse width.

PPA Configuration

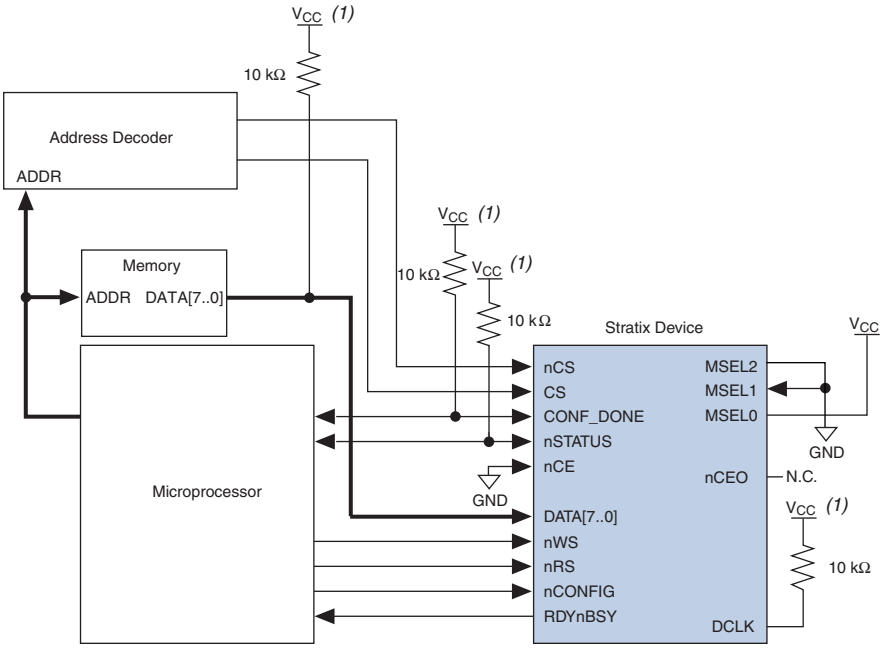
In PPA schemes, a microprocessor drives data to the Stratix or Stratix GX device through a download cable. When using a PPA scheme, use a 1-kΩ pull-up resistor to pull the DCLK pin high to prevent unused configuration pins from floating.

To begin configuration, the microprocessor drives nCONFIG high and then asserts the target device's nCS pin low and CS pin high. Next, the microprocessor places an 8-bit configuration word on the target device's data inputs and pulses nWS low. On the rising edge of nWS, the target device latches a byte of configuration data and then drives its RDYnBSY signal low, indicating that it is processing the byte of configuration data. The microprocessor then performs other system functions while the Stratix or Stratix GX device is processing the byte of configuration data.

Next, the microprocessor checks nSTATUS and CONF_DONE. If nSTATUS is high and CONF_DONE is low, the microprocessor sends the next data byte. If nSTATUS is low, the device is signaling an error and the microprocessor should restart configuration. However, if nSTATUS is high and all the configuration data is received, the device is ready for initialization. At the beginning of initialization, CONF_DONE goes high to indicate that configuration is complete. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize. Initialization, by default, uses an internal oscillator, which runs at 10 MHz. After initialization, this internal oscillator is turned off. When initialization is complete, the Stratix or Stratix GX device enters user mode.

Figure 11–16 shows the PPA configuration circuit. An optional address decoder controls the device's nCS and CS pins. This decoder allows the microprocessor to select the Stratix or Stratix GX device by accessing a particular address, simplifying the configuration process.

Figure 11–16. PPA Configuration Circuit



Note to Figure 11–16:
(1) The pull-up resistor should be connected to the same supply voltage as the Stratix or Stratix GX device.

The device's nCS or CS pins can be toggled during PPA configuration if the design meets the specifications for t_{CSSU}, t_{WSP}, and t_{CSH} given in Table 11–10 on page 11–36. The microprocessor can also directly control the nCS and CS signals. You can tie one of the nCS or CS signals to its active state (i.e., nCS may be tied low) and toggle the other signal to control configuration.

Stratix and Stratix GX devices can serialize data internally without the microprocessor. When the Stratix or Stratix GX device is ready for the next byte of configuration data, it drives RDYnBSY high. If the microprocessor senses a high signal when it polls RDYnBSY, the microprocessor strobes the next byte of configuration data into the device. Alternatively, the nRS signal can be strobed, causing the RDYnBSY signal to appear on DATA7. Because RDYnBSY does not need to

be monitored, reading the state of the configuration data by strobing nRS low saves a system I/O port. Do not drive data onto the data bus while nRS is low because it causes contention on DATA7. If the nRS pin is not used to monitor configuration, you should tie it high. To simplify configuration, the microprocessor can wait for the total time of t_{BUSY} (max) + t_{RDY2WS} + t_{W2SB} before sending the next data bit.

After configuration, the nCS, CS, nRS, nWS, and RDYnBSY pins act as user I/O pins. However, if the PPA scheme is chosen in the Quartus II software, these I/O pins are tri-stated by default in user mode and should be driven by the microprocessor. To change the default settings in the Quartus II software, select **Device & Pin Option** (Compiler Setting menu).

If the Stratix or Stratix GX device detects an error during configuration, it drives nSTATUS low to alert the microprocessor. The microprocessor can then pulse nCONFIG low to restart the configuration process. Alternatively, if the **Auto-Restart Configuration on Frame Error** option is turned on, the Stratix or Stratix GX device releases nSTATUS after a reset time-out period. After nSTATUS is released, the microprocessor can reconfigure the Stratix or Stratix GX device. At this point, the microprocessor does not need to pulse nCONFIG low.

The microprocessor can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The microprocessor must monitor the nSTATUS pin to detect errors and the CONF_DONE pin to determine when programming completes (CONF_DONE goes high one byte early in parallel mode). If the microprocessor sends all configuration data and starts initialization but CONF_DONE is not asserted, the microprocessor must reconfigure the Stratix or Stratix GX device.

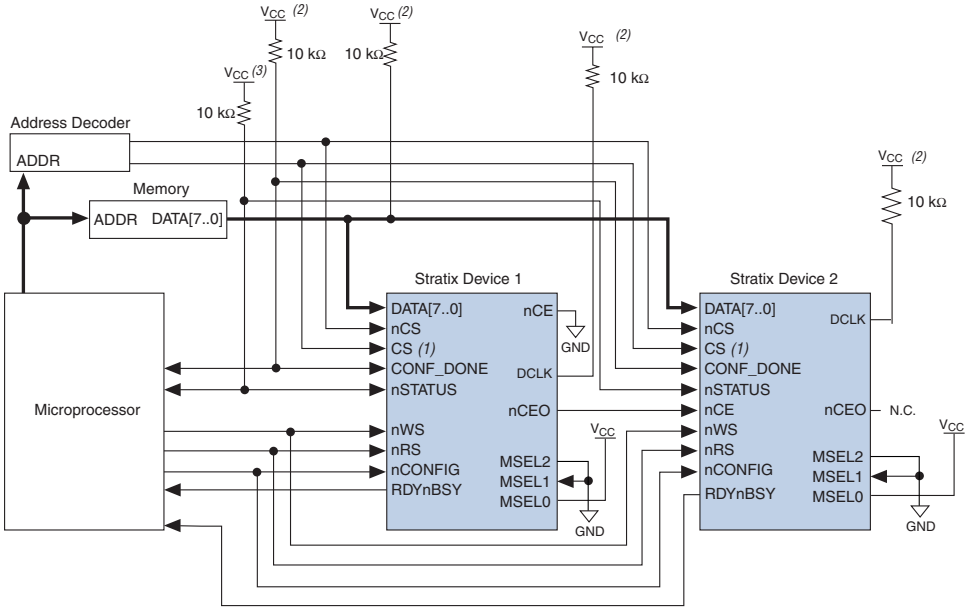
By default, the INIT_DONE is disabled. You can enable the INIT_DONE output by turning on the **Enable INIT_DONE output** option in the Quartus II software. If you do not turn on the **Enable INIT_DONE output** option in the Quartus II software, you are advised to wait for the maximum value of t_{CD2UM} (see Table 11–10) after the CONF_DONE signal goes high to ensure the device has been initialized properly and that it has entered user mode.

During configuration and initialization, and before the device enters user mode, the microprocessor must not drive the CONF_DONE signal low.

If the optional CLKUSR pin is used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure that CLKUSR continues toggling during the time nSTATUS is low (maximum of 40 µs).

You can also use PPA mode to configure multiple Stratix and Stratix GX devices. Multi-device PPA configuration is similar to single-device PPA configuration, except that the Stratix and Stratix GX devices are cascaded. After you configure the first Stratix or Stratix GX device, nCEO is asserted, which asserts the nCE pin on the second device, initiating configuration. Because the second Stratix or Stratix GX device begins configuration within one write cycle of the first device, the transfer of data destinations is transparent to the microprocessor. All Stratix and Stratix GX device CONF_DONE pins are tied together; therefore, all devices initialize and enter user mode at the same time. See Figure 11–17.

Figure 11–17. PPA Multi-Device Configuration Circuit



Notes to Figure 11–17:

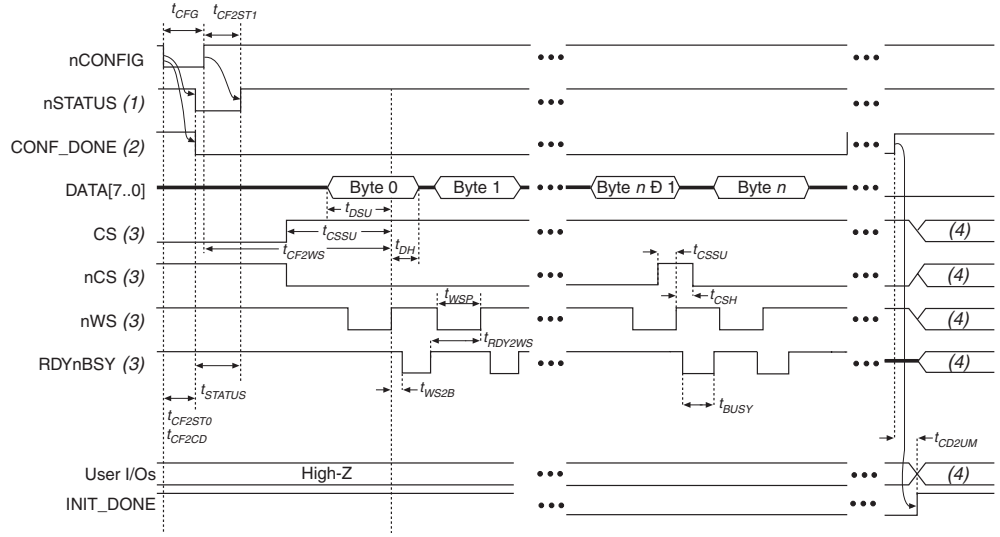
(1) If not used, you can connect the CS pin to V_{CC} directly. If not used, the nCS pin can be connected to GND directly.

(2) Connect the pull-up resistor to the same supply voltage as the Stratix or Stratix GX device.

PPA Configuration Timing

Figure 11–18 shows the Stratix and Stratix GX device timing waveforms for PPA configuration.

Figure 11–18. PPA Timing Waveforms for Stratix & Stratix GX Devices

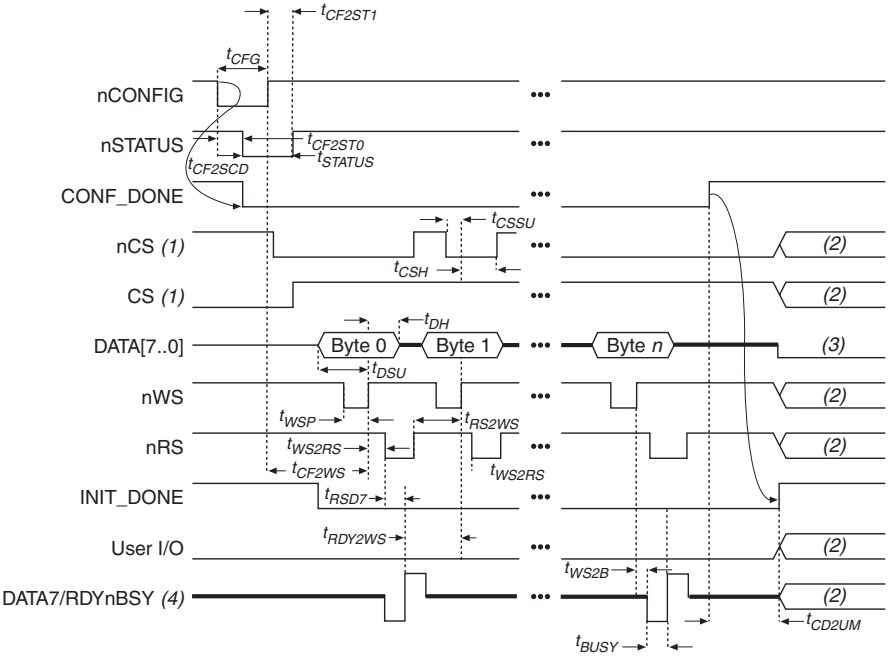


Notes to Figure 11–18:

(1) Upon power-up, nSTATUS is held low for the time of the POR delay.
(2) Upon power-up, before and during configuration, CONF_DONE is low.
(3) After configuration, the state of CS, nCS, nWS, and RDYnBSY depends on the design programmed into the Stratix or Stratix GX device.
(4) Device I/O pins are in user mode.

Figure 11–19 shows the Stratix and Stratix GX timing waveforms when using strobed nRS and nWS signals.

Figure 11–19. PPA Timing Waveforms Using Strobed nRS & nWS Signals



Notes to Figure 11–19:

(1) The user can toggle nCS or CS during configuration if the design meets the specification for t_{CSSU}, t_{WSP}, and t_{CSH}.

(2) Device I/O pins are in user mode.

(3) The DATA[7..0] pins are available as user I/Os after configuration and the state of theses pins depends on the dual-purpose pin settings. Do not leave DATA[7..0] floating. If these pins are not used in user-mode, you should drive them high or low, whichever is more convenient.

(4) DATA7 is a bidirectional pin. It represents an input for data input, but represents an output to show the status of RDYnBSY.

Table 11–10 defines the Stratix and Stratix GX timing parameters for PPA configuration

Table 11–10. PPA Timing Parameters for Stratix & Stratix GX Devices

Symbol	Parameter	Min	Max	Units
t_{CF2WS}	nCONFIG high to first rising edge on nWS	40		µs
t_{DSU}	Data setup time before rising edge on nWS	10		ns
t_{DH}	Data hold time after rising edge on nWS	0		ns
t_{CSSU}	Chip select setup time before rising edge on nWS	10		ns
t_{CSH}	Chip select hold time after rising edge on nWS	0		ns
t_{WSP}	nWS low pulse width	15		ns
t_{CFG}	nCONFIG low pulse width	40		µs
t_{WS2B}	nWS rising edge to RDYnBSY low		20	ns
t_{BUSY}	RDYnBSY low pulse width	7	45	ns
t_{RDY2WS}	RDYnBSY rising edge to nWS rising edge	15		ns
t_{WS2RS}	nWS rising edge to nRS falling edge	15		ns
t_{RS2WS}	nRS rising edge to nWS rising edge	15		ns
t_{RSD7}	nRS falling edge to DATA7 valid with RDYnBSY signal		20	ns
t_{CD2UM}	CONF_DONE high to user mode *(1)*	6	20	µs
t_{STATUS}	nSTATUS low pulse width	10	40 *(2)*	µs
t_{CF2CD}	nCONFIG low to CONF_DONE low		800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		800	ns
t_{CF2ST1}	nCONFIG high to nSTATUS high		40 *(2)*	µs

Notes to Table 11–10:

(1) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device. If the clock source is CLKUSR, multiply the clock period by 136 to obtain this value.

(2) This value is obtained if you do not delay configuration by extending the nstatus to low pulse width.

For information on how to create configuration and programming files for this configuration scheme, see the *Software Settings* section in the *Configuration Handbook, Volume 2*.

JTAG Programming & Configuration

The JTAG has developed a specification for boundary-scan testing. This boundary-scan test (BST) architecture offers the capability to efficiently test components on printed circuit boards (PCBs) with tight lead spacing. The BST architecture can test pin connections without using physical test

probes and capture functional data while a device is operating normally. You can also use the JTAG circuitry to shift configuration data into the device.

For more information on JTAG boundary-scan testing, see *AN 39: IEEE 1149.1 (JTAG) Boundary-Scan Testing in Altera Devices*.

To use the SignalTap® II embedded logic analyzer, you need to connect the JTAG pins of your Stratix device to a download cable header on your PCB.

For more information on SignalTap II, see the *Design Debugging Using SignalTap II Embedded Logic Analyzer* chapter in the *Quartus II Handbook, Volume 2*.

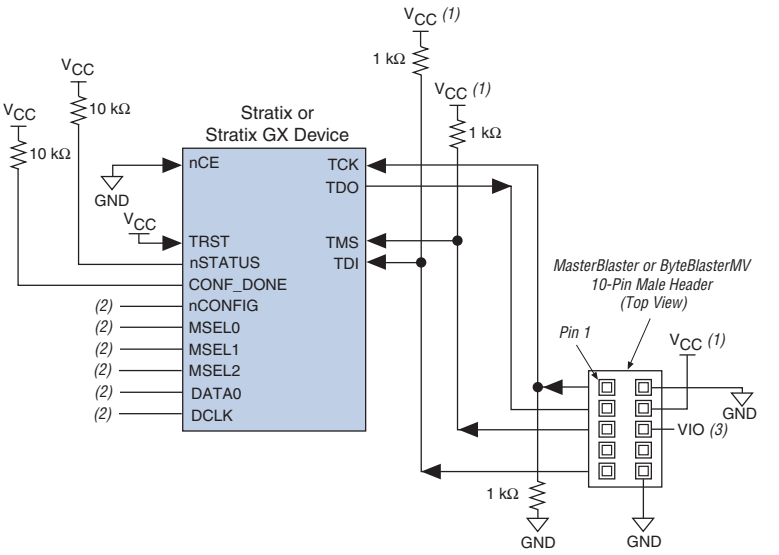
A device operating in JTAG mode uses four required pins, TDI, TDO, TMS, and TCK, and one optional pin, TRST. The four JTAG input pins (TDI, TMS, TCK and TRST) have weak, internal pull-up resistors, whose values range from 20 to 40 kΩ All other pins are tri-stated during JTAG configuration. Do not begin JTAG configuration until all other configuration is complete. Table 11–11 shows each JTAG pin's function.

Table 11–11. JTAG Pin Descriptions

Pin	Description	Function
TDI	Test data input	Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. The VCCSEL pin controls the input buffer selection.
TDO	Test data output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. The high level output voltage is determined by VCCIO.
TMS	Test mode select	Input pin that provides the control signal to determine the transitions of the Test Access Port (TAP) controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. The VCCSEL pin controls the input buffer selection.
TCK	Test clock input	The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. The VCCSEL pin controls the input buffer selection.
TRST	Test reset input (optional)	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to IEEE Std. 1149.1. The VCCSEL pin controls the input buffer selection.

During JTAG configuration, data is downloaded to the device on the PCB through the MasterBlaster or ByteBlasterMV header. Configuring devices through a cable is similar to programming devices in-system. One difference is to connect the TRST pin to V_{CC} to ensure that the TAP controller is not reset. See Figure 11–20.

Figure 11–20. JTAG Configuration of a Single Device



Notes to Figure 11–20:

(1) You should connect the pull-up resistor to the same supply voltage as the download cable.

(2) You should connect the nCONFIG, MSEL0, and MSEL1 pins to support a non-JTAG configuration scheme. If you only use JTAG configuration, connect nCONFIG to V_{CC}, and MSEL0, MSEL1, and MSEL2 to ground. Pull DATA0 and DCLK to high or low.

(3) V_{IO} is a reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. See the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value.

To configure a single device in a JTAG chain, the programming software places all other devices in BYPASS mode. In BYPASS mode, devices pass programming data from the TDI pin to the TDO pin through a single bypass register without being affected internally. This scheme enables the programming software to program or verify the target device. Configuration data driven into the device appears on the TDO pin one clock cycle later.

Stratix and Stratix GX devices have dedicated JTAG pins. You can perform JTAG testing on Stratix and Stratix GX devices before and after, but not during configuration. The chip-wide reset and output enable pins on Stratix and Stratix GX devices do not affect JTAG boundary-scan or programming operations. Toggling these pins does not affect JTAG operations (other than the usual boundary-scan operation).

When designing a board for JTAG configuration of Stratix and Stratix GX devices, you should consider the regular configuration pins. Table 11–12 shows how you should connect these pins during JTAG configuration.

Table 11–12. Dedicated Configuration Pin Connections During JTAG Configuration

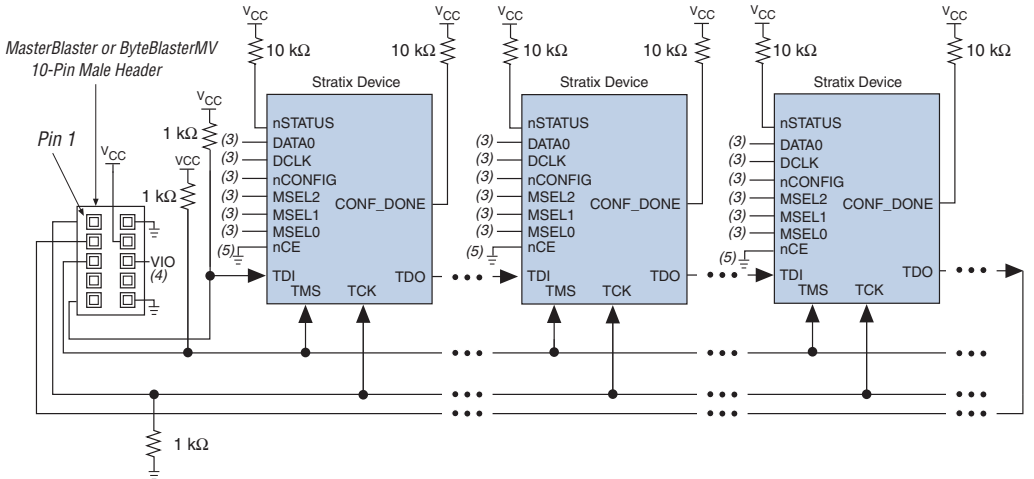
Signal	Description
nCE	On all Stratix and Stratix GX devices in the chain, nCE should be driven low by connecting it to ground, pulling it low via a resistor, or driving it by some control circuitry. For devices that are also in multi-device PS, FPP or PPA configuration chains, the nCE pins should be connected to GND during JTAG configuration or JTAG configured in the same order as the configuration chain.
nCEO	On all Stratix and Stratix GX devices in the chain, nCEO can be left floating or connected to the nCE of the next device. See nCE pin description above.
MSEL	These pins must not be left floating. These pins support whichever non-JTAG configuration is used in production. If only JTAG configuration is used, you should tie both pins to ground.
nCONFIG	nCONFIG must be driven high through the JTAG programming process. Driven high by connecting to V_{CC}, pulling high via a resistor, or driven by some control circuitry.
nSTATUS	Pull to V_{CC} via a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each nSTATUS pin should be pulled up to V_{CC} individually. nSTATUS pulling low in the middle of JTAG configuration indicates that an error has occurred.
CONF_DONE	Pull to V_{CC} via a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each CONF_DONE pin should be pulled up to V_{CC} individually. CONF_DONE going high at the end of JTAG configuration indicates successful configuration.
DCLK	Should not be left floating. Drive low or high, whichever is more convenient on your board.
DATA0	Should not be left floating. Drive low or high, whichever is more convenient on your board.

JTAG Programming & Configuration of Multiple Devices

When programming a JTAG device chain, one JTAG-compatible header, such as the ByteBlasterMV header, is connected to several devices. The number of devices in the JTAG chain is limited only by the drive capacity of the download cable. However, when more than five devices are connected in a JTAG chain, Altera recommends buffering the TCK, TDI, and TMS pins with an on-board buffer.

JTAG-chain device programming is ideal when the PCB contains multiple devices, or when testing the PCB using JTAG BST circuitry. Figure 11–21 shows multi-device JTAG configuration.

***Figure 11–21. Multi-Device JTAG Configuration** Notes (1), (2)*



Notes to Figure 11–21:

(1) Stratix, Stratix GX, APEX™ II, APEX 20K, Mercury™, ACEX® 1K, and FLEX® 10K devices can be placed within the same JTAG chain for device programming and configuration.

(2) For more information on all configuration pins connected in this mode, see Table 11–11 on page 11–37.

(3) Connect the nCONFIG, MSEL0, MSEL1, and MSEL2 pins to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect nCONFIG to V_{CC}, and MSEL0, MSEL1, and MSEL2 to ground. Pull DATA0 and DCLK to either high or low.

(4) V_{IO} is a reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. See the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value.

(5) nCE must be connected to GND or driven low for successful JTAG configuration.

The nCE pin must be connected to GND or driven low during JTAG configuration. In multi-device PS, FPP and PPA configuration chains, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. Therefore, if these devices are also in a JTAG chain, you should make sure the nCE pins are connected to GND during JTAG configuration or that the devices are JTAG configured in the same order as the configuration chain. As long as the devices are JTAG configured in the same order as the multi-device configuration chain, the nCEO of the previous device drives nCE of the next device low when it has successfully been JTAG configured.

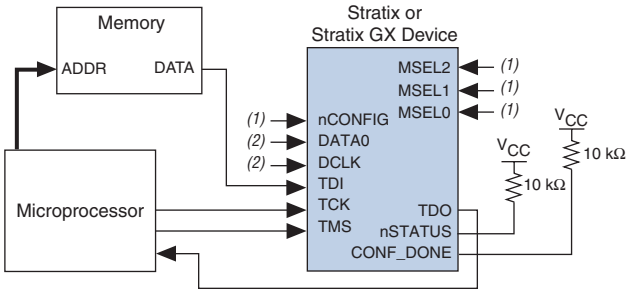
The Quartus II software verifies successful JTAG configuration upon completion. The software checks the state of CONF_DONE through the JTAG port. If CONF_DONE is not in the correct state, the Quartus II software indicates that configuration has failed. If CONF_DONE is in the correct state, the software indicates that configuration was successful.

If VCCIO is tied to 3.3 V, both the I/O pins and JTAG TDO port drive at 3.3-V levels.

Do not attempt JTAG and non-JTAG configuration simultaneously. When configuring through JTAG, allow any non-JTAG configuration to complete first.

Figure 11–22 shows the JTAG configuration of a Stratix or Stratix GX device with a microprocessor.

Figure 11–22. JTAG Configuration of Stratix & Stratix GX Devices with a Microprocessor



Notes to Figure 11–22:

(1) Connect the nCONFIG, MSEL2, MSEL1, and MSEL0 pins to support a non-JTAG configuration scheme. If your design only uses JTAG configuration, connect the nCONFIG pin to V_{CC} and the MSEL2, MSEL1, and MSEL0 pins to ground.

(2) Pull DATA0 and DCLK to either high or low.

Configuration with JRunner Software Driver

JRunner is a software driver that allows you to configure Altera FPGAs through the ByteBlasterMV download cable in JTAG mode. The programming input file supported is in Raw Binary File (**.rbf**) format. JRunner also requires a Chain Description File (**.cdf**) generated by the Quartus II software. JRunner is targeted for embedded JTAG configuration. The source code has been developed for the Windows NT operating system. You can customize the code to make it run on other platforms.

For more information on the JRunner software driver, see the *JRunner Software Driver: An Embedded Solution to the JTAG Configuration White Paper* and zip file.

Jam STAPL Programming & Test Language

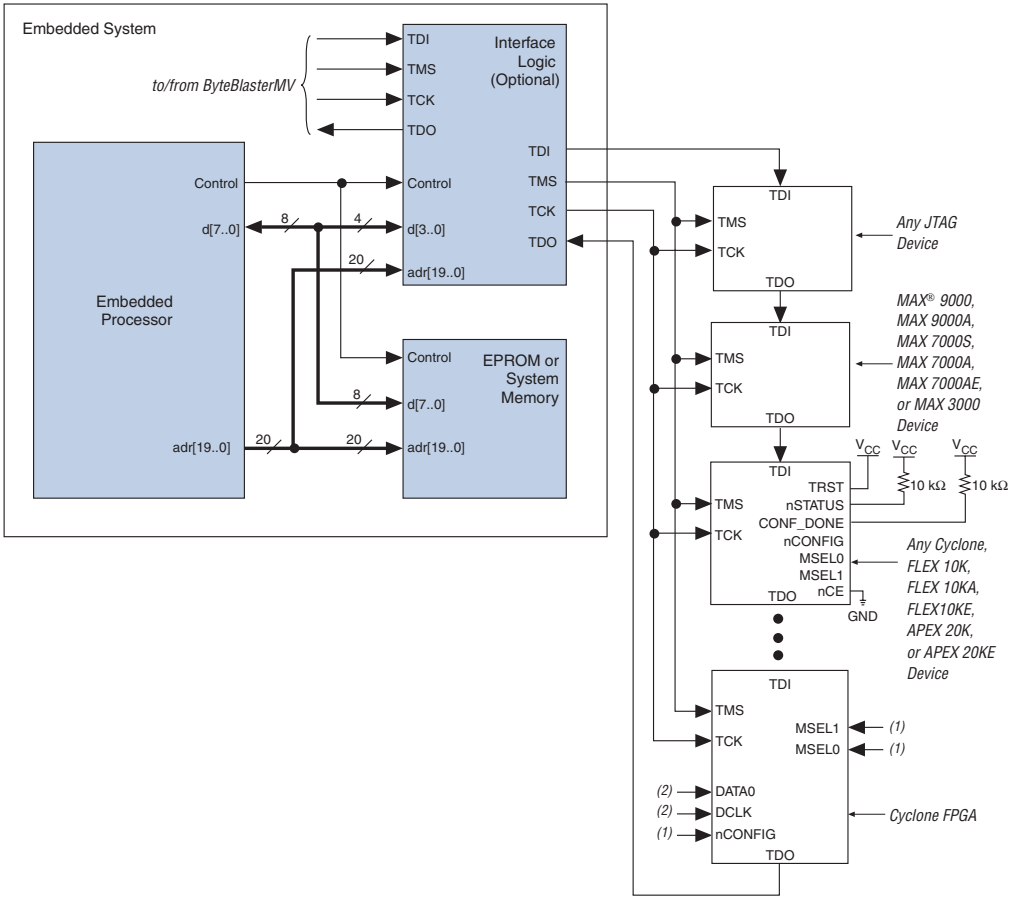
The Jam™ Standard Test and Programming Language (STAPL), JEDEC standard JESD-71, is a standard file format for in-system programmability (ISP) purposes. Jam STAPL supports programming or configuration of programmable devices and testing of electronic systems, using the IEEE 1149.1 JTAG interface. Jam STAPL is a freely licensed open standard.

Connecting the JTAG Chain to the Embedded Processor

There are two ways to connect the JTAG chain to the embedded processor. The most straightforward method is to connect the embedded processor directly to the JTAG chain. In this method, four of the processor pins are dedicated to the JTAG interface, saving board space but reducing the number of available embedded processor pins.

Figure 11–23 illustrates the second method, which is to connect the JTAG chain to an existing bus through an interface PLD. In this method, the JTAG chain becomes an address on the existing bus. The processor then reads from or writes to the address representing the JTAG chain.

Figure 11–23. Embedded System Block Diagram



Notes to Figure 11–23:

(1) Connect the nCONFIG, MSEL2, MSEL1, and MSEL0 pins to support a non-JTAG configuration scheme. If your design only uses JTAG configuration, connect the nCONFIG pin to V_{CC} and the MSEL2, MSEL1, and MSEL0 pins to ground.

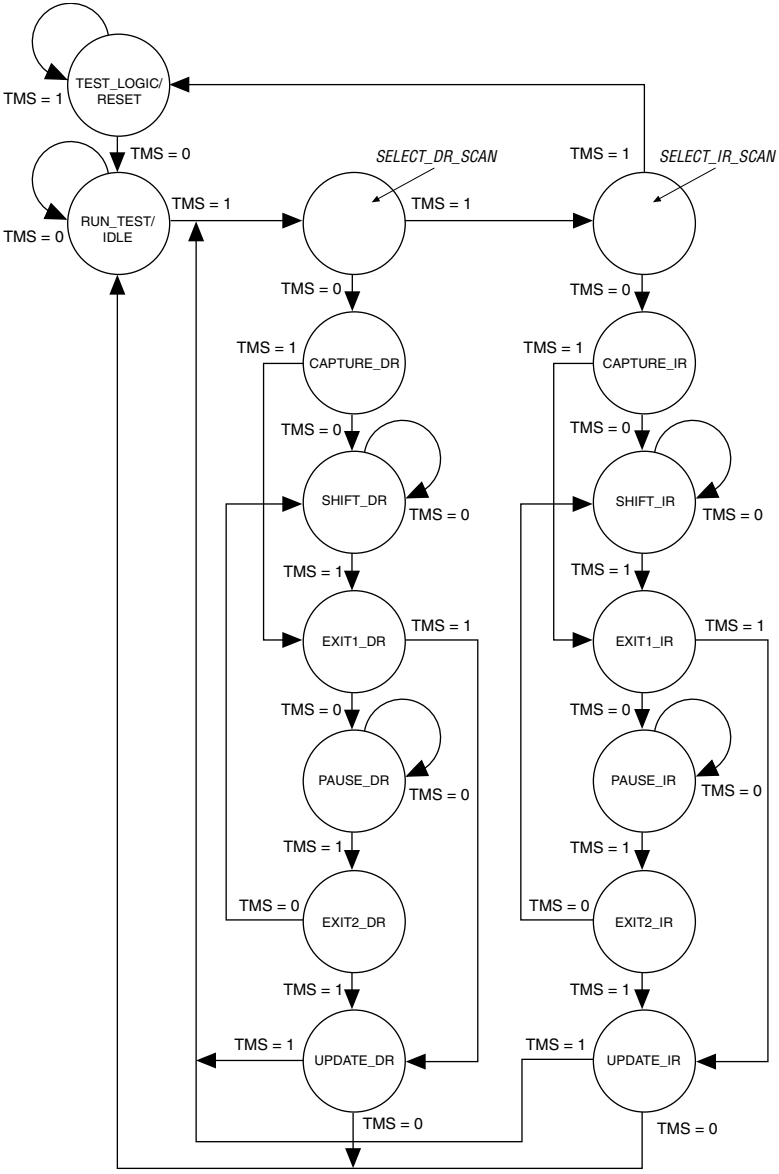
(2) Pull DATA0 and DCLK to either high or low.

Both JTAG connection methods should include space for the MasterBlaster or ByteBlasterMV header connection. The header is useful during prototyping because it allows you to verify or modify the Stratix or Stratix GX device's contents. During production, you can remove the header to save cost.

Program Flow

The Jam Player provides an interface for manipulating the IEEE Std. 1149.1 JTAG TAP state machine. The TAP controller is a 16-state state machine that is clocked on the rising edge of TCK, and uses the TMS pin to control JTAG operation in a device. Figure 11–24 shows the flow of an IEEE Std. 1149.1 TAP controller state machine.

Figure 11–24. JTAG TAP Controller State Machine



While the Jam Player provides a driver that manipulates the TAP controller, the Jam Byte-Code File (**.jbc**) provides the high-level intelligence needed to program a given device. All Jam instructions that

send JTAG data to the device involve moving the TAP controller through either the data register leg or the instruction register leg of the state machine. For example, loading a JTAG instruction involves moving the TAP controller to the SHIFT_IR state and shifting the instruction into the instruction register through the TDI pin. Next, the TAP controller is moved to the RUN_TEST/IDLE state where a delay is implemented to allow the instruction time to be latched. This process is identical for data register scans, except that the data register leg of the state machine is traversed.

The high-level Jam instructions are the DRSCAN instruction for scanning the JTAG data register, the IRSCAN instruction for scanning the instruction register, and the WAIT command that causes the state machine to sit idle for a specified period of time. Each leg of the TAP controller is scanned repeatedly, according to instructions in the JBC file, until all of the target devices are programmed.

Figure 11–25 illustrates the functional behavior of the Jam Player when it parses the JBC file. When the Jam Player encounters a DRSCAN, IRSCAN, or WAIT instruction, it generates the proper data on TCK, TMS, and TDI to complete the instruction. The flow diagram shows branches for the DRSCAN, IRSCAN, and WAIT instructions. Although the Jam Player supports other instructions, they are omitted from the flow diagram for simplicity.

Figure 11–25. Jam Player Flow Diagram (Part 1 of 2)

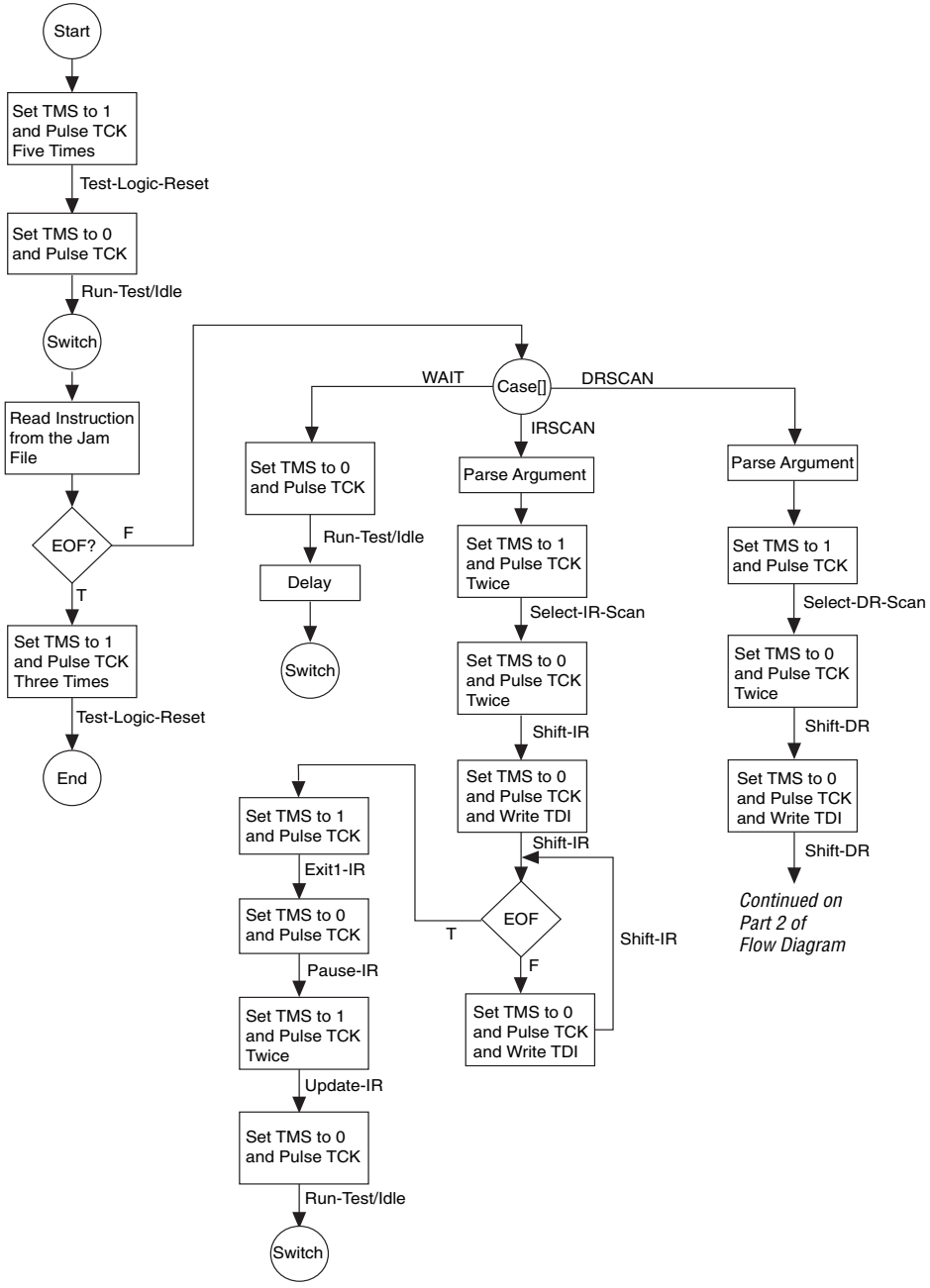
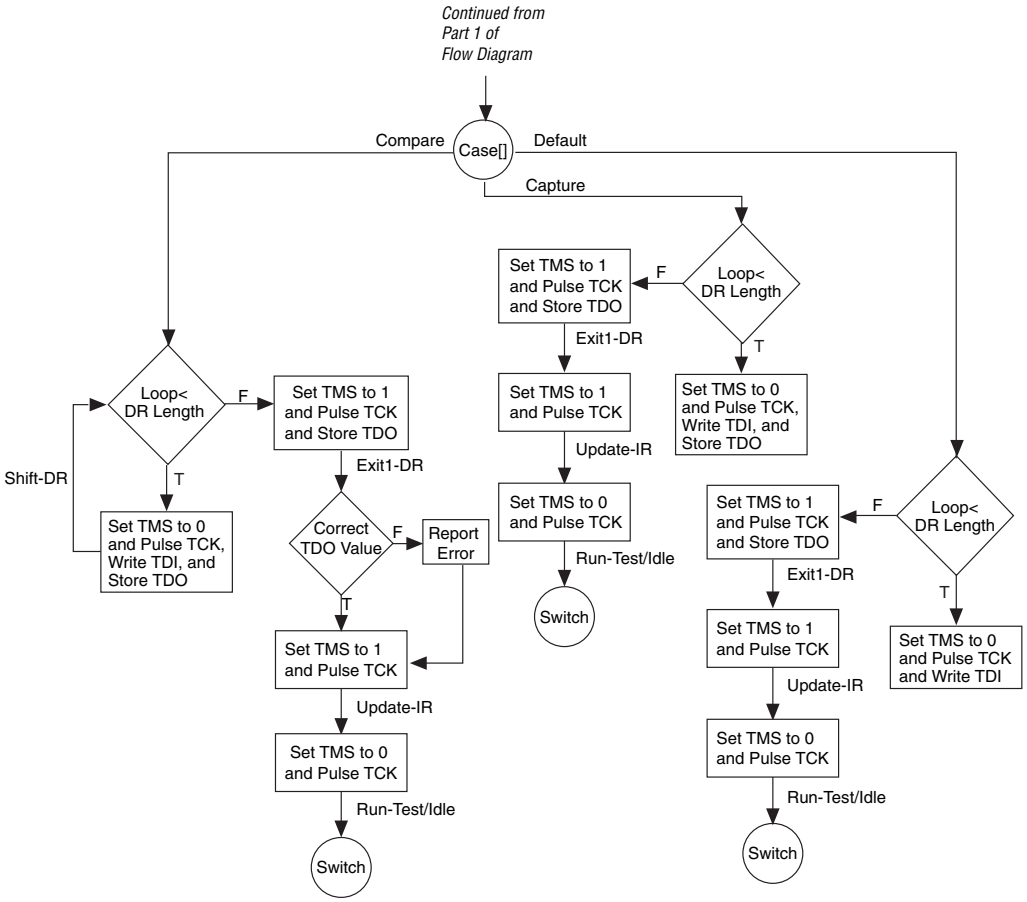


Figure 11–26. Jam Player Flow Diagram (Part 2 of 2)


Continued from Part 1 of Flow Diagram
Compare
Case[]
Default
Capture
Loop< DR Length
F
T
Shift-DR
Set TMS to 1 and Pulse TCK and Store TDO
Exit1-DR
Set TMS to 0 and Pulse TCK, Write TDI, and Store TDO
Correct TDO Value
Report Error
Set TMS to 1 and Pulse TCK
Update-IR
Set TMS to 0 and Pulse TCK
Run-Test/Idle
Switch
Set TMS to 0 and Pulse TCK, Write TDI, and Store TDO
Set TMS to 1 and Pulse TCK
Set TMS to 0 and Pulse TCK and Write TDI

Execution of a Jam program starts at the beginning of the program. The program flow is controlled using GOTO, CALL/RETURN, and FOR/NEXT structures. The GOTO and CALL statements see labels that are symbolic names for program statements located elsewhere in the Jam program. The language itself enforces almost no constraints on the organizational structure or control flow of a program.

The Jam language does not support linking multiple Jam programs together or including the contents of another file into a Jam program.

Jam Instructions

Each Jam statement begins with one of the instruction names listed in Table 11–13. The instruction names, including the names of the optional instructions, are reserved keywords that you cannot use as variable or label identifiers in a Jam program.

Table 11–13. Instruction Names

BOOLEAN	INTEGER	PREIR
CALL	IRSCAN	PRINT
CRC	IRSTOP	PUSH
DRSCAN	LET	RETURN
DRSTOP	NEXT	STATE
EXIT	NOTE	WAIT
EXPORT	POP	VECTOR *(1)*
FOR	POSTDR	VMAP *(1)*
GOTO	POSTIR	–
IF	PREDR	–

Note to Table 11–13:
(1) This instruction name is an optional language extension.

Table 11–14 shows the state names that are reserved keywords in the Jam language. These keywords correspond to the state names specified in the IEEE Std. 1149.1 JTAG specification.

Table 11–14. Reserved Keywords (Part 1 of 2)

IEEE Std. 1149.1 JTAG State Names	Jam Reserved State Names
Test-Logic-Reset	RESET
Run-Test-Idle	IDLE
Select-DR-Scan	DRSELECT
Capture-DR	DRCAPTURE
Shift-DR	DRSHIFT
Exit1-DR	DREXIT1
Pause-DR	DRPAUSE
Exit2-DR	DREXIT2
Update-DR	DRUPDATE
Select-IR-Scan	IRSELECT

Table 11–14. Reserved Keywords (Part 2 of 2)	
IEEE Std. 1149.1 JTAG State Names	**Jam Reserved State Names**
Capture-IR	IRCAPTURE
Shift-IR	IRSHIFT
Exit1-IR	IREXIT1
Pause-IR	IRPAUSE
Exit2-IR	IREXIT2
Update-IR	IRUPDATE

Example Jam File that Reads the IDCODE

Figure 11–27 illustrates the flexibility and utility of the Jam STAPL. The example reads the IDCODE out of a single device in a JTAG chain.

The array variable, I_IDCODE, is initialized with the IDCODE instruction bits ordered the LSB first (on the left) to most significant bit (MSB) (on the right). This order is important because the array field in the IRSCAN instruction is always interpreted, and sent, MSB to LSB.

Figure 11–27. Example Jam File Reading IDCODE

```
BOOLEAN read_data[32];
BOOLEAN I_IDCODE[10] = BIN 1001101000; 'assumed
BOOLEAN ONES_DATA[32] = HEX FFFFFFFF;
INTEGER i;
'Set up stop state for IRSCAN
IRSTOP IRPAUSE;
'Initialize device
STATE RESET;
IRSCAN 10, I_IDCODE[0..9]; 'LOAD IDCODE INSTRUCTION
STATE IDLE;
WAIT 5 USEC, 3 CYCLES;
DRSCAN 32, ONES_DATA[0..31], CAPTURE
read_data[0..31];
'CAPTURE IDCODE
PRINT "IDCODE:";
FOR i=0 to 31;
PRINT read_data[i];
NEXT i;
EXIT 0;
```

Configuring Using the MicroBlaster Driver

The MicroBlaster™ software driver allows you to configure Altera devices in an embedded environment using PS or FPP mode. The MicroBlaster software driver supports a Raw Binary File (**.rbf**) programming input file. The source code is developed for the Windows NT operating system, although you can customize it to run on other operating systems. For more information on the MicroBlaster software driver, go to the Altera web site (**www.altera.com**).

Device Configuration Pins

The following tables describe the connections and functionality of all the configuration related pins on the Stratix or Stratix GX device. Table 11–15 describes the dedicated configuration pins, which are required to be connected properly on your board for successful configuration. Some of these pins may not be required for your configuration schemes.

Table 11–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 1 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
VCCSEL	N/A	All	Input	Dedicated input that selects which input buffer is used on the configuration input pins; nCONFIG, DCLK, RUnLU, nCE, nWS, nRS, CS, nCS and CLKUSR. The VCCSEL input buffer is powered by V_{CCINT} and has an internal 2.5 kΩ pull-down resistor that is always active. A logic high (1.5-V, 1.8-V, 2.5-V, 3.3-V) selects the 1.8-V/1.5-V input buffer, and a logic low selects the 3.3-V/2.5-V input buffer. See the "V_{CCSEL} Pins" section for more details.
PORSEL	N/A	All	Input	Dedicated input which selects between a POR time of 2 ms or 100 ms. A logic high (1.5-V, 1.8-V, 2.5-V, 3.3-V) selects a POR time of about 2 ms and a logic low selects POR time of about 100 ms. The PORSEL input buffer is powered by V_{CCINT} and has an internal 2.5 kΩ pull-down resistor that is always active.

Table 11–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 2 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nIO_PULLUP	N/A	All	Input	Dedicated input that chooses whether the internal pull-ups on the user I/Os and dual-purpose I/Os (DATA[7..0], nWS, nRS, RDYnBSY, nCS, CS, RUnLU, PGM[], CLKUSR, INIT_DONE, DEV_OE, DEV_CLR) are on or off before and during configuration. A logic high (1.5-V, 1.8-V, 2.5-V, 3.3-V) turns off the weak internal pull-ups, while a logic low turns them on. The nIO_PULLUP input buffer is powered by V_{CCINT} and has an internal 2.5 kΩ pull-down resistor that is always active.
MSEL[2..0]	N/A	All	Input	3-bit configuration input that sets the Stratix or Stratix GX device configuration scheme. See Table 11–2 for the appropriate connections. These pins can be connected to V_{CCIO} of the I/O bank they reside in or ground. This pin uses Schmitt trigger input buffers.
nCONFIG	N/A	All	Input	Configuration control input. Pulling this pin low during user-mode causes the FPGA to lose its configuration data, enter a reset state, tri-state all I/O pins. Returning this pin to a logic high level initiates a reconfiguration. If your configuration scheme uses an enhanced configuration device or EPC2 device, nCONFIG can be tied directly to V_{CC} or to the configuration device's nINIT_CONF pin. This pin uses Schmitt trigger input buffers.

Table 11–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 3 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nSTATUS	N/A	All	Bidirectional open-drain	The device drives nSTATUS low immediately after power-up and releases it after the POR time. Status output. If an error occurs during configuration, nSTATUS is pulled low by the target device. Status input. If an external source drives the nSTATUS pin low during configuration or initialization, the target device enters an error state. Driving nSTATUS low after configuration and initialization does not affect the configured device. If a configuration device is used, driving nSTATUS low causes the configuration device to attempt to configure the FPGA, but since the FPGA ignores transitions on nSTATUS in user-mode, the FPGA does not reconfigure. To initiate a reconfiguration, nCONFIG must be pulled low. The enhanced configuration devices' and EPC2 devices' OE and nCS pins have optional internal programmable pull-up resistors. If internal pull-up resistors on the enhanced configuration device are used, external 10-kΩ pull-up resistors should not be used on these pins. When using EPC2 devices, only external 10-kΩ pull-up resistors should be used. This pin uses Schmitt trigger input buffers.

Table 11–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 4 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
CONF_DONE	N/A	All	Bidirectional open-drain	Status output. The target FPGA drives the CONF_DONE pin low before and during configuration. Once all configuration data is received without error and the initialization cycle starts, the target device releases CONF_DONE. Status input. After all data is received and CONF_DONE goes high, the target device initializes and enters user mode. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize. Driving CONF_DONE low after configuration and initialization does not affect the configured device. The enhanced configuration devices' and EPC2 devices' OE and nCS pins have optional internal programmable pull-up resistors. If internal pull-up resistors on the enhanced configuration device are used, external 10-kΩ pull-up resistors should not be used on these pins. When using EPC2 devices, only external 10-kΩ pull-up resistors should be used. This pin uses Schmitt trigger input buffers.
nCE	N/A	All	Input	Active-low chip enable. The nCE pin activates the device with a low signal to allow configuration. The nCE pin must be held low during configuration, initialization, and user mode. In single device configuration, it should be tied low. In multi-device configuration, nCE of the first device is tied low while its nCEO pin is connected to nCE of the next device in the chain. The nCE pin must also be held low for successful JTAG programming of the FPGA. This pin uses Schmitt trigger input buffers.

Table 11–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 5 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nCEO	N/A	All Multi-Device Schemes	Output	Output that drives low when device configuration is complete. In single device configuration, this pin is left floating. In multi-device configuration, this pin feeds the next device's nCE pin. The nCEO of the last device in the chain is left floating. The voltage levels driven out by this pin are dependent on the V_{CCIO} of the I/O bank it resides in.
DCLK	N/A	Synchronous configuration schemes (PS, FPP)	Input (PS, FPP)	In PS and FPP configuration, DCLK is the clock input used to clock data from an external source into the target device. Data is latched into the FPGA on the rising edge of DCLK. In PPA mode, DCLK should be tied high to V_{CC} to prevent this pin from floating. After configuration, this pin is tri-stated. In schemes that use a configuration device, DCLK is driven low after configuration is done. In schemes that use a control host, DCLK should be driven either high or low, whichever is more convenient. Toggling this pin after configuration does not affect the configured device. This pin uses Schmitt trigger input buffers.
DATA0	I/O	PS, FPP, PPA	Input	Data input. In serial configuration modes, bit-wide configuration data is presented to the target device on the DATA0 pin. The V_{IH} and V_{IL} levels for this pin are dependent on the V_{CCIO} of the I/O bank that it resides in. After configuration, DATA0 is available as a user I/O and the state of this pin depends on the **Dual-Purpose Pin** settings. After configuration, EPC1 and EPC1441 devices tri-state this pin, while enhanced configuration and EPC2 devices drive this pin high.

Table 11–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 6 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
DATA[7..1]	I/O	Parallel configuration schemes (FPP and PPA)	Inputs	Data inputs. Byte-wide configuration data is presented to the target device on DATA[7..0]. The V_{IH} and V_{IL} levels for these pins are dependent on the V_{CCIO} of the I/O banks that they reside in. In serial configuration schemes, they function as user I/Os during configuration, which means they are tri-stated. After PPA or FPP configuration, DATA[7..1] are available as a user I/Os and the state of these pin depends on the **Dual-Purpose Pin** settings.
DATA7	I/O	PPA	Bidirectional	In the PPA configuration scheme, the DATA7 pin presents the RDYnBSY signal after the nRS signal has been strobed low. The V_{IL} and V_{IL} levels for this pin are dependent on the V_{CCIO} of the I/O bank that it resides in. In serial configuration schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, DATA7 is available as a user I/O and the state of this pin depends on the **Dual-Purpose Pin** settings.
nWS	I/O	PPA	Input	Write strobe input. A low-to-high transition causes the device to latch a byte of data on the DATA[7..0] pins. In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, nWS is available as a user I/O and the state of this pin depends on the **Dual-Purpose Pin** settings.

Table 11–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 7 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nRS	I/O	PPA	Input	Read strobe input. A low input directs the device to drive the RDYnBSY signal on the DATA7 pin. If the nRS pin is not used in PPA mode, it should be tied high. In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, nRS is available as a user I/O and the state of this pin depends on the **Dual-Purpose Pin** settings.
RDYnBSY	I/O	PPA	Output	Ready output. A high output indicates that the target device is ready to accept another data byte. A low output indicates that the target device is busy and not ready to receive another data byte. In PPA configuration schemes, this pin drives out high after power-up, before configuration and after configuration before entering user-mode. In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, RDYnBSY is available as a user I/O and the state of this pin depends on the **Dual-Purpose Pin** settings.

Table 11–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 8 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nCS/CS	I/O	PPA	Input	Chip-select inputs. A low on nCS and a high on CS select the target device for configuration. The nCS and CS pins must be held active during configuration and initialization. During the PPA configuration mode, it is only required to use either the nCS or CS pin. Therefore, if only one chip-select input is used, the other must be tied to the active state. For example, nCS can be tied to GND while CS is toggled to control configuration.In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, nCS and CS are available as a user I/Os and the state of these pins depends on the **Dual-Purpose Pin** settings.
RUnLU	N/A if using Remote Configuration; I/O if not	Remote Configuration in FPP, PS or PPA	Input	Input that selects between remote update and local update. A logic high (1.5-V, 1.8-V, 2.5-V, 3.3-V) selects remote update and a logic low selects local update. When not using remote update or local update configuration modes, this pins is available as general-purpose user I/O pin.
PGM[2..0]	N/A if using Remote Configuration; I/O if not using	Remote Configuration in FPP, PS or PPA	Input	These output pins select one of eight pages in the memory (either flash or enhanced configuration device) when using a remote configuration mode. When not using remote update or local update configuration modes, these pins are available as general-purpose user I/O pins.

Table 11–16 describes the optional configuration pins. If these optional configuration pins are not enabled in the Quartus II software, they are available as general-purpose user I/O pins. Therefore during configuration, these pins function as user I/O pins and are tri-stated with weak pull-ups.

Table 11–16. Optional Configuration Pins

Pin Name	User Mode	Pin Type	Description
CLKUSR	N/A if option is on. I/O if option is off.	Input	Optional user-supplied clock input. Synchronizes the initialization of one or more devices. This pin is enabled by turning on the **Enable user-supplied start-up clock (CLKUSR)** option in the Quartus II software.
INIT_DONE	N/A if option is on. I/O if option is off.	Output open-drain	Status pin. Can be used to indicate when the device has initialized and is in user mode. When nCONFIG is low and during the beginning of configuration, the INIT_DONE pin is tri-stated and pulled high due to an external 10-kΩ pull-up. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high and the FPGA enters user mode. Thus, the monitoring circuitry must be able to detect a low-to-high transition. This pin is enabled by turning on the **Enable INIT_DONE output** option in the Quartus II software.
DEV_OE	N/A if option is on. I/O if option is off.	Input	Optional pin that allows the user to override all tri-states on the device. When this pin is driven low, all I/Os are tri-stated. When this pin is driven high, all I/Os behave as programmed. This pin is enabled by turning on the **Enable device-wide output enable (DEV_OE)** option in the Quartus II software.
DEV_CLRn	N/A if option is on. I/O if option is off.	Input	Optional pin that allows you to override all clears on all device registers. When this pin is driven low, all registers are cleared. When this pin is driven high, all registers behave as programmed. This pin is enabled by turning on the **Enable device-wide reset (DEV_CLRn)** option in the Quartus II software.

Table 11–17 describes the dedicated JTAG pins. JTAG pins must be kept stable before and during configuration to prevent accidental loading of JTAG instructions. If you plan to use the SignalTap II Embedded Logic Analyzer, you will need to connect the JTAG pins of your device to a JTAG header on your board.

Table 11–17. Dedicated JTAG pins

Pin Name	User Mode	Pin Type	Description
TDI	N/A	Input	Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}. This pin uses Schmitt trigger input buffers.
TDO	N/A	Output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by leaving this pin unconnected.
TMS	N/A	Input	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}. This pin uses Schmitt trigger input buffers.
TCK	N/A	Input	The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND. This pin uses Schmitt trigger input buffers.
TRST	N/A	Input	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to IEEE Std. 1149.1. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND. This pin uses Schmitt trigger input buffers.

8. Configuration and Remote System Upgrades in Cyclone IV Devices

CYIV-51008-1.3

This chapter describes the configuration and remote system upgrades in Cyclone® IV devices. Cyclone IV (Cyclone IV GX and Cyclone IV E) devices use SRAM cells to store configuration data. You must download the configuration data to Cyclone IV devices each time the device powers up because SRAM memory is volatile.

Cyclone IV devices are configured using one of the following configuration schemes:

- Active serial (AS)
- Active parallel (AP) (supported in Cyclone IV E devices only)
- Passive serial (PS)
- Fast passive parallel (FPP) (not supported in EP4CGX15, EP4CGX22, and EP4CGX30 [except for the F484 package] devices)
- JTAG

Cyclone IV devices offer the following configuration features:

- Configuration data decompression ("Configuration Data Decompression" on page 8–2)
- Remote system upgrade ("Remote System Upgrade" on page 8–68)

System designers face difficult challenges, such as shortened design cycles, evolving standards, and system deployments in remote locations. Cyclone IV devices help overcome these challenges with inherent re-programmability and dedicated circuitry to perform remote system upgrades. Remote system upgrades help deliver feature enhancements and bug fixes without costly recalls, reduced time-to-market, and extended product life.

Configuration

This section describes Cyclone IV device configuration and includes the following topics:

- "Configuration Features" on page 8–2
- "Configuration Requirement" on page 8–3
- "Configuration Process" on page 8–6
- "Configuration Scheme" on page 8–8
- "AS Configuration (Serial Configuration Devices)" on page 8–10
- "AP Configuration (Supported Flash Memories)" on page 8–20
- "PS Configuration" on page 8–31
- "FPP Configuration" on page 8–39
- "JTAG Configuration" on page 8–44
- "Device Configuration Pins" on page 8–61

Configuration Features

Table 8–1 lists the configuration methods you can use in each configuration scheme.

Table 8–1. Configuration Features in Cyclone IV Devices

Configuration Scheme	Configuration Method	Decompression	Remote System Upgrade *(1)*
AS	Serial Configuration Device	✓	✓
AP	Supported Flash Memory *(2)*	—	✓
PS	External Host with Flash Memory	✓	✓*(3)*
	Download Cable	✓	—
FPP	External Host with Flash Memory	—	✓*(3)*
JTAG based configuration	External Host with Flash Memory	—	—
	Download Cable	—	—

Notes to Table 8–1:

(1) Remote update mode is supported when you use the Remote System Upgrade feature. You can enable or disable remote update mode with an option setting in the Quartus® II software.

(2) For more information about the supported device families for the Numonyx commodity parallel flash, refer to Table 8–8 on page 8–21.

(3) Remote update mode is supported externally using the Parallel Flash Loader (PFL) with the Quartus II software.

Configuration Data Decompression

Cyclone IV devices support configuration data decompression, which saves configuration memory space and time. This feature allows you to store compressed configuration data in configuration devices or other memory and send the compressed bitstream to Cyclone IV devices. During configuration, Cyclone IV devices decompress the bitstream in real time and program the SRAM cells.

Preliminary data indicates that compression reduces the configuration bitstream size by 35 to 55%.

When you enable compression, the Quartus II software generates configuration files with compressed configuration data. This compressed file reduces the storage requirements in the configuration device or flash memory and decreases the time required to send the bitstream to the Cyclone IV device. The time required by a Cyclone IV device to decompress a configuration file is less than the time required to send the configuration data to the device. There are two methods for enabling compression for the Cyclone IV device bitstreams in the Quartus II software:

- Before design compilation (through the Compiler Settings menu)
- After design compilation (through the **Convert Programming Files** dialog box)

To enable compression in the compiler settings of the project in the Quartus II software, perform the following steps:

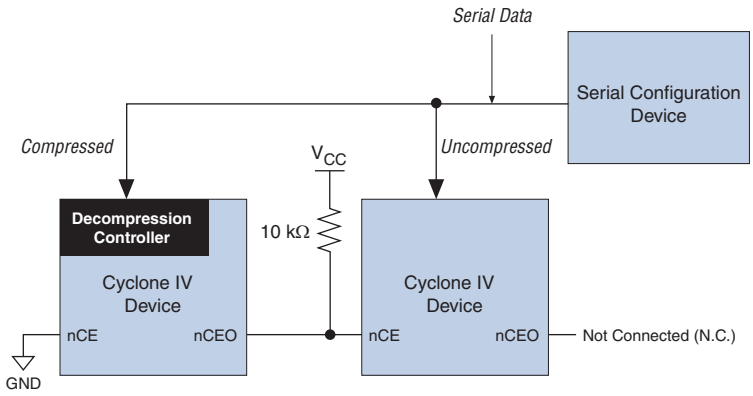
1. On the Assignments menu, click **Device**. The **Settings** dialog box appears.
2. Click **Device and Pin Options**. The **Device and Pin Options** dialog box appears.
3. Click the **Configuration** tab.
4. Turn on **Generate compressed bitstreams**.
5. Click **OK**.
6. In the **Settings** dialog box, click **OK**.

You can enable compression when creating programming files from the **Convert Programming Files** dialog box. To enable compression, perform the following steps:

1. On the File menu, click **Convert Programming Files**.
2. Under **Output programming file**, select your desired file type from the **Programming file type** list.
3. If you select **Programmer Object File (.pof)**, you must specify the configuration device in the **Configuration device** list.
4. Under **Input files to convert**, select **SOF Data**.
5. Click **Add File** to browse to the Cyclone IV device SRAM object files (**.sof)**.
6. In the **Convert Programming Files** dialog box, select the **.pof** you added to **SOF Data** and click **Properties**.
7. In the **SOF File Properties** dialog box, turn on the **Compression** option.

When multiple Cyclone IV devices are cascaded, you can selectively enable the compression feature for each device in the chain. Figure 8–1 shows a chain of two Cyclone IV devices. The first device has compression enabled and receives compressed bitstream from the configuration device. The second device has the compression feature disabled and receives uncompressed data. You can generate programming files for this setup in the **Convert Programming Files** dialog box.

Figure 8–1. Compressed and Uncompressed Configuration Data in the Same Configuration File



Configuration Requirement

This section describes Cyclone IV device configuration requirement and includes the following topics:

- "Power-On Reset (POR) Circuit" on page 8–4
- "Configuration File Size" on page 8–4
- "Power Up" on page 8–6

Power-On Reset (POR) Circuit

The POR circuit keeps the device in reset state until the power supply voltage levels have stabilized during device power up. After device power up, the device does not release nSTATUS until V_{CCINT}, V_{CCA}, and V_{CCIO} (for I/O banks in which the configuration and JTAG pins reside) are above the POR trip point of the device. V_{CCINT} and V_{CCA} are monitored for brown-out conditions after device power up.

V_{CCA} is the analog power to the phase-locked loop (PLL).

In some applications, it is necessary for a device to wake up very quickly to begin operation. Cyclone IV devices offer the fast POR time option to support fast wake-up time applications. The fast POR time option has stricter power-up requirements when compared with the standard POR time option. You can select either the fast option or the standard POR option with the MSEL pin settings.

If your system exceeds the fast or standard POR time, you must hold nCONFIG low until all the power supplies are stable.

For more information about the POR specifications, refer to the *Cyclone IV Device Datasheet*.

For more information about the wake-up time and POR circuit, refer to the *Power Requirements for Cyclone IV Devices* chapter.

Configuration File Size

Table 8–2 lists the approximate uncompressed configuration file sizes for Cyclone IV devices. To calculate the amount of storage space required for multiple device configurations, add the file size of each device together.

Table 8–2. Uncompressed Raw Binary File (.rbf) Sizes for Cyclone IV Devices (Part 1 of 2)

Device		Data Size (bits)
Cyclone IV E	EP4CE6	2,944,088
	EP4CE10	2,944,088
	EP4CE15	4,086,848
	EP4CE22	5,748,552
	EP4CE30	9,534,304
	EP4CE40	9,534,304
	EP4CE55	14,889,560
	EP4CE75	19,965,752
	EP4CE115	28,571,696

Table 8–2. Uncompressed Raw Binary File (.rbf) Sizes for Cyclone IV Devices (Part 2 of 2)

Device		Data Size (bits)
Cyclone IV GX	EP4CGX15	3,805,568
	EP4CGX22	7,600,040
	EP4CGX30	7,600,040
		24,500,000 *(1)*, *(2)*
	EP4CGX50	24,500,000 *(1)*
	EP4CGX75	24,500,000 *(1)*
	EP4CGX110	39,425,016
	EP4CGX150	39,425,016

Notes to Table 8–2:

(1) These values are preliminary.

(2) Only for the F484 package.

Use the data in Table 8–2 to estimate the file size before design compilation. Different configuration file formats, such as Hexadecimal (**.hex**) or Tabular Text File (**.ttf**) formats, have different file sizes. However, for any specific version of the Quartus II software, any design targeted for the same device has the same uncompressed configuration file size. If you use compression, the file size varies after each compilation, because the compression ratio depends on the design.

For more information about setting device configuration options or creating configuration files, refer to the *Software Settings* section in volume 2 of the *Configuration Handbook*.

Configuration and JTAG Pin I/O Requirements

Cyclone IV devices are manufactured using the TSMC 60-nm low-k dielectric process. Although Cyclone IV devices use TSMC 2.5-V transistor technology in the I/O buffers, the devices are compatible and able to interface with 2.5, 3.0, and 3.3-V configuration voltage standards by following specific requirements.

All I/O inputs must maintain a maximum AC voltage of 4.1 V. When using a serial configuration device in an AS configuration scheme, you must connect a 25-Ω series resistor for the `DATA[0]` pin. When cascading the Cyclone IV device family in a multi-device configuration for AS, AP, FPP, and PS configuration schemes, you must connect the repeater buffers between the master and slave devices for the `DATA` and `DCLK` pins. When using the JTAG configuration scheme in a multi-device configuration, connect 25-Ω resistors on both ends of the `TDO-TDI` path if the `TDO` output driver is a non-Cyclone IV device.

The output resistance of the repeater buffers and the `TDO` path for all cases must fit the maximum overshoot equation shown in Equation 8–1.

Equation 8–1. *(Note 1)*

$$0.8Z_O \leq R_E \leq 1.8Z_O$$

Note to Equation 8–1:

(1) Z_O is the transmission line impedance and R_E is the equivalent resistance of the output buffer.

Configuration Process

This section describes Cyclone IV device configuration requirements and includes the following topics:

- "Power Up" on page 8–6
- "Reset" on page 8–6
- "Configuration" on page 8–6
- "Configuration Error" on page 8–7
- "Initialization" on page 8–7
- "User Mode" on page 8–7

For more information about the Altera® FPGA configuration cycle state machine, refer to the *Configuring Altera FPGAs* chapter in volume 1 of the *Configuration Handbook*.

Power Up

If the device is powered up from the power-down state, V_{CCINT}, V_{CCA}, and V_{CCIO} (for the I/O banks in which the configuration and JTAG pins reside) must be powered up to the appropriate level for the device to exit from POR.

Reset

After power up, Cyclone IV devices go through POR. POR delay depends on the MSEL pin settings, which correspond to your configuration scheme. During POR, the device resets, holds nSTATUS and CONF_DONE low, and tri-states all user I/O pins (for PS and FPP configuration schemes only). To tri-state the configuration bus for AS and AP configuration schemes, you must tie nCE high. The user I/O pins and dual-purpose I/O pins have weak pull-up resistors, which are always enabled (after POR) before and during configuration. When the device exits POR, all user I/O pins continue to tri-state. While nCONFIG is low, the device is in reset. When nCONFIG goes high, the device exits reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. After nSTATUS is released, the device is ready to receive configuration data and the configuration stage starts.

For more information about the value of the weak pull-up resistors on the I/O pins that are on before and during configuration, refer to the *Cyclone IV Device Datasheet* chapter.

Configuration

Configuration data is latched into the Cyclone IV device at each DCLK cycle. However, the width of the data bus and the configuration time taken for each scheme are different. After the device receives all the configuration data, the device releases the open-drain CONF_DONE pin, which is pulled high by an external 10-kΩ pull-up resistor. A low-to-high transition on the CONF_DONE pin indicates that the configuration is complete and initialization of the device can begin.

You can begin reconfiguration by pulling the nCONFIG pin low. The nCONFIG pin must be low for at least 500 ns. When nCONFIG is pulled low, the Cyclone IV device is reset. The Cyclone IV device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. When nCONFIG returns to a logic-high level and nSTATUS is released by the Cyclone IV device, reconfiguration begins.

Configuration Error

If an error occurs during configuration, Cyclone IV devices assert the nSTATUS signal low, indicating a data frame error and the CONF_DONE signal stays low. If the **Auto-restart configuration after error** option (available in the Quartus II software in the **General** tab of the **Device and Pin Options** dialog box) is turned on, the Cyclone IV device releases nSTATUS after a reset time-out period (a maximum of 230 µs), and retries configuration. If this option is turned off, the system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 500 ns to restart configuration.

Initialization

In Cyclone IV devices, the initialization clock source is either the internal oscillator or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If you use the internal oscillator, the device provides itself with enough clock cycles for proper initialization. When using the internal oscillator, you do not have to send additional clock cycles from an external source to the CLKUSR pin during the initialization stage. Additionally, you can use the CLKUSR pin as a user I/O pin.

You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the **CLKUSR** option. The CLKUSR pin allows you to control when your device enters user mode for an indefinite amount of time. You can turn on the **Enable user-supplied start-up clock (CLKUSR)** option in the Quartus II software in the **General** tab of the **Device and Pin Options** dialog box. When you turn on the **Enable user supplied start-up clock option (CLKUSR)** option, the CLKUSR pin is the initialization clock source. Supplying a clock on the CLKUSR pin does not affect the configuration process. After the configuration data is accepted and CONF_DONE goes high, Cyclone IV devices require 3,192 clock cycles to initialize properly and enter user mode.

If you use the optional CLKUSR pin and the nCONFIG pin is pulled low to restart configuration during device initialization, ensure that the CLKUSR pin continues to toggle when nSTATUS is low (a maximum of 230 µs).

User Mode

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user mode with a low-to-high transition. The **Enable INIT_DONE Output** option is available in the Quartus II software in the **General** tab of the **Device and Pin Options** dialog box. If you use the INIT_DONE pin, it is high due to an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. After the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. This low-to-high transition signals that the device has entered user mode. In user mode, the user I/O pins function as assigned in your design and no longer have weak pull-up resistors.

Configuration Scheme

A configuration scheme with different configuration voltage standards is selected by driving the MSEL pins either high or low, as shown in Table 8–3, Table 8–4, and Table 8–5.

Hardwire the MSEL pins to V_{CCA} or GND without pull-up or pull-down resistors to avoid problems detecting an incorrect configuration scheme. Do not drive the MSEL pins with a microprocessor or another device.

Table 8–3. Configuration Schemes for Cyclone IV GX Devices (EP4CGX15, EP4CGX22, and EP4CGX30 [except for F484 Package])

Configuration Scheme	MSEL2	MSEL1	MSEL0	POR Delay	Configuration Voltage Standard (V) *(1)*
AS	1	0	1	Fast	3.3
	0	1	1	Fast	3.0, 2.5
	0	0	1	Standard	3.3
	0	1	0	Standard	3.0, 2.5
PS	1	0	0	Fast	3.3, 3.0, 2.5
	1	1	0	Fast	1.8, 1.5
	0	0	0	Standard	3.3, 3.0, 2.5
JTAG-based configuration *(2)*	*(3)*	*(3)*	*(3)*	—	—

Notes to Table 8–3:

(1) Configuration voltage standard applied to the V_{CCIO} supply of the bank in which the configuration pins reside.

(2) JTAG-based configuration takes precedence over other configuration schemes, which means the MSEL pin settings are ignored.

(3) Do not leave the MSEL pins floating. Connect them to V_{CCA} or GND. These pins support the non-JTAG configuration scheme used in production. Altera recommends connecting the MSEL pins to GND if your device is only using JTAG configuration.

Table 8–4. Configuration Schemes for Cyclone IV GX Devices (EP4CGX30 [only for F484 package], EP4CGX50, EP4CGX75, EP4CGX110, and EP4CGX150) (Part 1 of 2)

Configuration Scheme	MSEL3	MSEL2	MSEL1	MSEL0	POR Delay	Configuration Voltage Standard (V) *(1)*
AS	1	1	0	1	Fast	3.3
	1	0	1	1	Fast	3.0, 2.5
	1	0	0	1	Standard	3.3
	1	0	1	0	Standard	3.0, 2.5
PS	1	1	0	0	Fast	3.3, 3.0, 2.5
	1	1	1	0	Fast	1.8, 1.5
	1	0	0	0	Standard	3.3, 3.0, 2.5
	0	0	0	0	Standard	1.8, 1.5
FPP	0	0	1	1	Fast	3.3, 3.0, 2.5
	0	1	0	0	Fast	1.8, 1.5
	0	0	0	1	Standard	3.3, 3.0, 2.5
	0	0	1	0	Standard	1.8, 1.5

Table 8–4. Configuration Schemes for Cyclone IV GX Devices (EP4CGX30 [only for F484 package], EP4CGX50, EP4CGX75, EP4CGX110, and EP4CGX150) (Part 2 of 2)

Configuration Scheme	MSEL3	MSEL2	MSEL1	MSEL0	POR Delay	Configuration Voltage Standard (V) *(1)*
JTAG-based configuration *(2)*	*(3)*	*(3)*	*(3)*	*(3)*	—	—

Notes to Table 8–4:

(1) Configuration voltage standard applied to the V_{CCIO} supply of the bank in which the configuration pins reside.

(2) JTAG-based configuration takes precedence over other configuration schemes, which means the MSEL pin settings are ignored.

(3) Do not leave the MSEL pins floating. Connect them to V_{CCA} or GND. These pins support the non-JTAG configuration scheme used in production. Altera recommends connecting the MSEL pins to GND if your device is only using JTAG configuration.

Smaller Cyclone IV E devices or package options (E144 and F256 packages) do not have the MSEL[3] pin. The AS Fast POR configuration scheme at 3.0- or 2.5-V configuration voltage standard and the AP configuration scheme are not supported in Cyclone IV E devices without the MSEL[3] pin. To configure these devices with other supported configuration schemes, select MSEL[2..0] pins according to the MSEL settings in Table 8–5.

Table 8–5. Configuration Schemes for Cyclone IV E Devices

Configuration Scheme	MSEL3	MSEL2	MSEL1	MSEL0	POR Delay	Configuration Voltage Standard (V) *(1)*
AS	1	1	0	1	Fast	3.3
	0	1	0	0	Fast	3.0, 2.5
	0	0	1	0	Standard	3.3
	0	0	1	1	Standard	3.0, 2.5
AP	0	1	0	1	Fast	3.3
	0	1	1	0	Fast	1.8
	0	1	1	1	Standard	3.3
	1	0	1	1	Standard	3.0, 2.5
	1	0	0	0	Standard	1.8
PS	1	1	0	0	Fast	3.3, 3.0, 2.5
	0	0	0	0	Standard	3.3, 3.0, 2.5
FPP	1	1	1	0	Fast	3.3, 3.0, 2.5
	1	1	1	1	Fast	1.8, 1.5
JTAG-based configuration *(2)*	*(3)*	*(3)*	*(3)*	*(3)*	—	—

Notes to Table 8–5:

(1) Configuration voltage standard applied to the V_{CCIO} supply of the bank in which the configuration pins reside.

(2) JTAG-based configuration takes precedence over other configuration schemes, which means the MSEL pin settings are ignored.

(3) Do not leave the MSEL pins floating. Connect them to V_{CCA} or GND. These pins support the non-JTAG configuration scheme used in production. Altera recommends connecting the MSEL pins to GND if your device is only using JTAG configuration.

For Cyclone IV E devices, the Quartus II software prohibits you from using the LVDS I/O standard in I/O Bank 1 when the configuration device I/O voltage is not 2.5 V. If you need to assign LVDS I/O standard in I/O Bank 1, navigate to **Assignments**>**Device**>**Settings**>**Device and Pin Option**>**Configuration** to change the Configuration Device I/O voltage to **2.5 V** or **Auto**.

AS Configuration (Serial Configuration Devices)

In the AS configuration scheme, Cyclone IV devices are configured with a serial configuration device. These configuration devices are low-cost devices with non-volatile memories that feature a simple four-pin interface and a small form factor. These features make serial configuration devices the ideal low-cost configuration solution.

For more information about serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Datasheet* in volume 2 of the *Configuration Handbook*.

Serial configuration devices provide a serial interface to access the configuration data. During device configuration, Cyclone IV devices read the configuration data through the serial interface, decompress the data if necessary, and configure their SRAM cells. This scheme is referred to as the AS configuration scheme because the device controls the configuration interface.

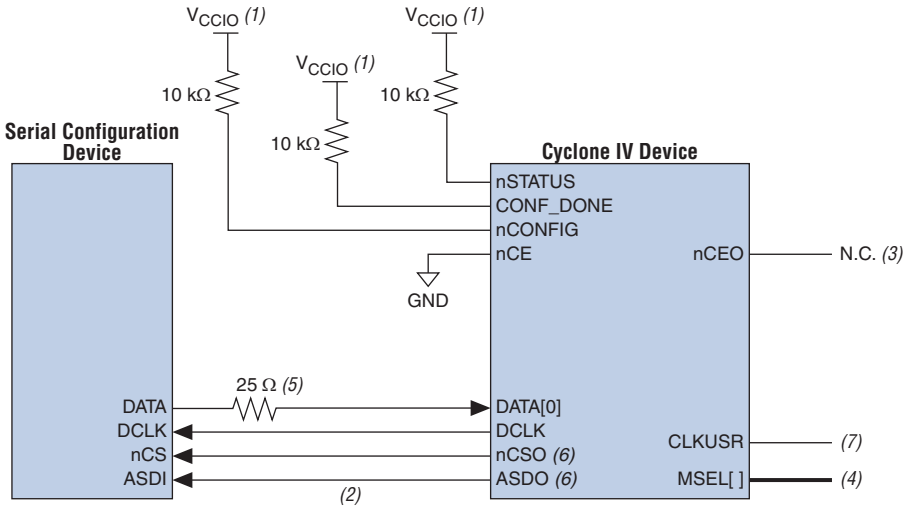
Single-Device AS Configuration

The four-pin interface of serial configuration devices consists of the following pins:

- Serial clock input (DCLK)
- Serial data output (DATA)
- Active-low chip select (nCS)
- AS data input (ASDI)

This four-pin interface connects to Cyclone IV device pins, as shown in Figure 8–2.

Figure 8–2. Single-Device AS Configuration



Notes to Figure 8–2:

(1) Connect the pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) Cyclone IV devices use the ASDO-to-ASDI path to control the configuration device.

(3) The nCEO pin is left unconnected or used as a user I/O pin when it does not feed the nCE pin of another device.

(4) The MSEL pin settings vary for different configuration voltage standards and POR time. To connect the MSEL pins, refer to Table 8–3 on page 8–8, Table 8–4 on page 8–8, and Table 8–5 on page 8–9. Connect the MSEL pins directly to V_{CCA} or GND.

(5) Connect the series resistor at the near end of the serial configuration device.

(6) These pins are dual-purpose I/O pins. The nCSO pin functions as FLASH_nCE pin in AP mode. The ASDO pin functions as the DATA[1] pin in AP and FPP modes.

(7) Only Cyclone IV GX devices have an option to select CLKUSR (40 MHz maximum) as the external clock source for DCLK.

The 25-Ω resistor at the near end of the serial configuration device for DATA[0] works to minimize the driver impedance mismatch with the board trace and reduce the overshoot seen at the Cyclone IV device DATA[0] input pin.

In the single-device AS configuration, the maximum board loading and board trace length between the supported serial configuration device and the Cyclone IV device must follow the recommendations in Table 8–6 on page 8–17.

The DCLK generated by the Cyclone IV device controls the entire configuration cycle and provides timing for the serial interface. Cyclone IV devices use an internal oscillator or an external clock source to generate the DCLK. For Cyclone IV E devices, you can use a 40-MHz internal oscillator to generate the DCLK and for Cyclone IV GX devices you can use a slow clock (20 MHz maximum) or a fast clock (40 MHz maximum) from the internal oscillator or an external clock from CLKUSR to generate the DCLK. There are some variations in the internal oscillator frequency because of the process, voltage, and temperature (PVT) conditions in Cyclone IV devices. The internal oscillator is designed to ensure that its maximum frequency is guaranteed to meet EPCS device specifications. Cyclone IV devices offer the option to select CLKUSR as the external clock source for DCLK. You can change the clock source option in the Quartus II software in the **Configuration** tab of the **Device and Pin Options** dialog box.

EPCS1 does not support Cyclone IV devices because of its insufficient memory capacity.

In configuration mode, the Cyclone IV device enables the serial configuration device by driving the nCSO output pin low, which connects to the nCS pin of the configuration device. The Cyclone IV device uses the DCLK and DATA[1] pins to send operation commands and read address signals to the serial configuration device. The configuration device provides data on its DATA pin, which connects to the DATA[0] input of the Cyclone IV device.

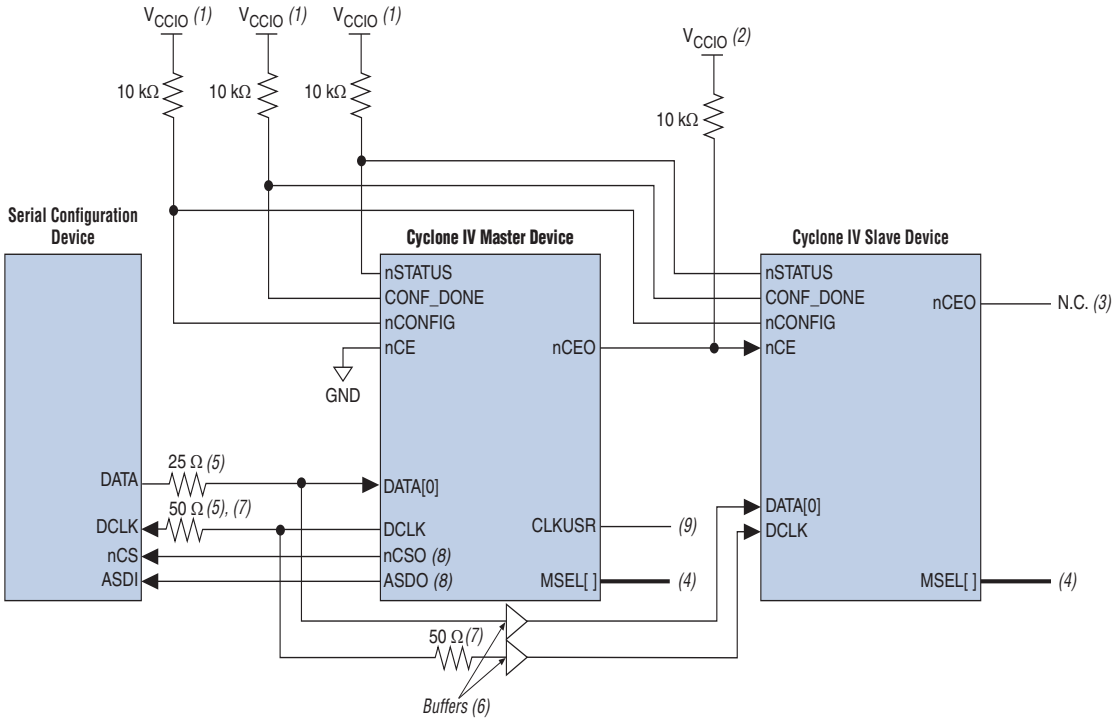
All AS configuration pins (DATA[0], DCLK, nCSO, and DATA[1]) have weak internal pull-up resistors that are always active. After configuration, these pins are set as input tri-stated and are driven high by the weak internal pull-up resistors.

The timing parameters for AS mode are not listed here because the t_{CF2CD}, t_{CF2ST0}, t_{CFG}, t_{STATUS}, t_{CF2ST1}, and t_{CD2UM} timing parameters are identical to the timing parameters for PS mode shown in Table 8–10 on page 8–35.

Multi-Device AS Configuration

You can configure multiple Cyclone IV devices with a single serial configuration device. When the first device captures all its configuration data from the bitstream, it drives the nCEO pin low, enabling the next device in the chain. If the last device in the chain is a Cyclone IV device, you can leave the nCEO pin of the last device unconnected or use it as a user I/O pin after configuration. The nCONFIG, nSTATUS, CONF_DONE, DCLK, and DATA[0] pins of each device in the chain are connected together (Figure 8–3).

Figure 8–3. Multi-Device AS Configuration



Notes to Figure 8–3:

(1) Connect the pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) Connect the pull-up resistor to the V_{CCIO} supply voltage of I/O bank in which the nCE pin resides.

(3) You can leave the nCEO pin unconnected or use it as a user I/O pin when it does not feed the nCE pin of another device.

(4) The MSEL pin settings vary for different configuration voltage standards and POR time. You must set the master device of the Cyclone IV device in AS mode and the slave devices in PS mode. To connect the MSEL pins for the master device in AS mode and slave devices in PS mode, refer to Table 8–3 on page 8–8, Table 8–4 on page 8–8, and Table 8–5 on page 8–9. Connect the MSEL pins directly to V_{CCA} or GND.

(5) Connect the series resistor at the near end of the serial configuration device.

(6) Connect the repeater buffers between the master and slave devices of the Cyclone IV device for DATA[0] and DCLK. All I/O inputs must maintain a maximum AC voltage of 4.1 V. The output resistance of the repeater buffers must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 8–5.

(7) The 50-Ω series resistors are optional if the 3.3-V configuration voltage standard is applied. For optimal signal integrity, connect these 50-Ω series resistors if the 2.5- or 3.0-V configuration voltage standard is applied.

(8) These pins are dual-purpose I/O pins. The nCSO pin functions as FLASH_nCE pin in AP mode. The ASDO pin functions as DATA[1] pin in AP and FPP modes.

(9) Only Cyclone IV GX devices have an option to select CLKUSR (40 MHz maximum) as the external clock source for DCLK.

The first Cyclone IV device in the chain is the configuration master and it controls the configuration of the entire chain. Other Altera devices that support PS configuration can also be part of the chain as configuration slaves.

☞ In the multi-device AS configuration, the board trace length between the serial configuration device and the master device of the Cyclone IV device must follow the recommendations in Table 8–6 on page 8–17.

The nSTATUS and CONF_DONE pins on all target devices are connected together with external pull-up resistors, as shown in Figure 8–3 on page 8–13. These pins are open-drain bidirectional pins on the devices. When the first device asserts nCEO (after receiving all its configuration data), it releases its CONF_DONE pin. However, the subsequent devices in the chain keep this shared CONF_DONE line low until they receive their configuration data. When all target devices in the chain receive their configuration data and release CONF_DONE, the pull-up resistor drives a high level on CONF_DONE line and all devices simultaneously enter initialization mode.

☞ Although you can cascade Cyclone IV devices, serial configuration devices cannot be cascaded or chained together.

If the configuration bitstream size exceeds the capacity of a serial configuration device, you must select a larger configuration device, enable the compression feature, or both. When configuring multiple devices, the size of the bitstream is the sum of the individual device's configuration bitstream.

Configuring Multiple Cyclone IV Devices with the Same Design

Certain designs require that you configure multiple Cyclone IV devices with the same design through a configuration bitstream, or a **.sof**. You can do this through the following methods:

- Multiple **.sof**
- Single **.sof**

☞ For both methods, the serial configuration devices cannot be cascaded or chained together.

Multiple SRAM Object Files

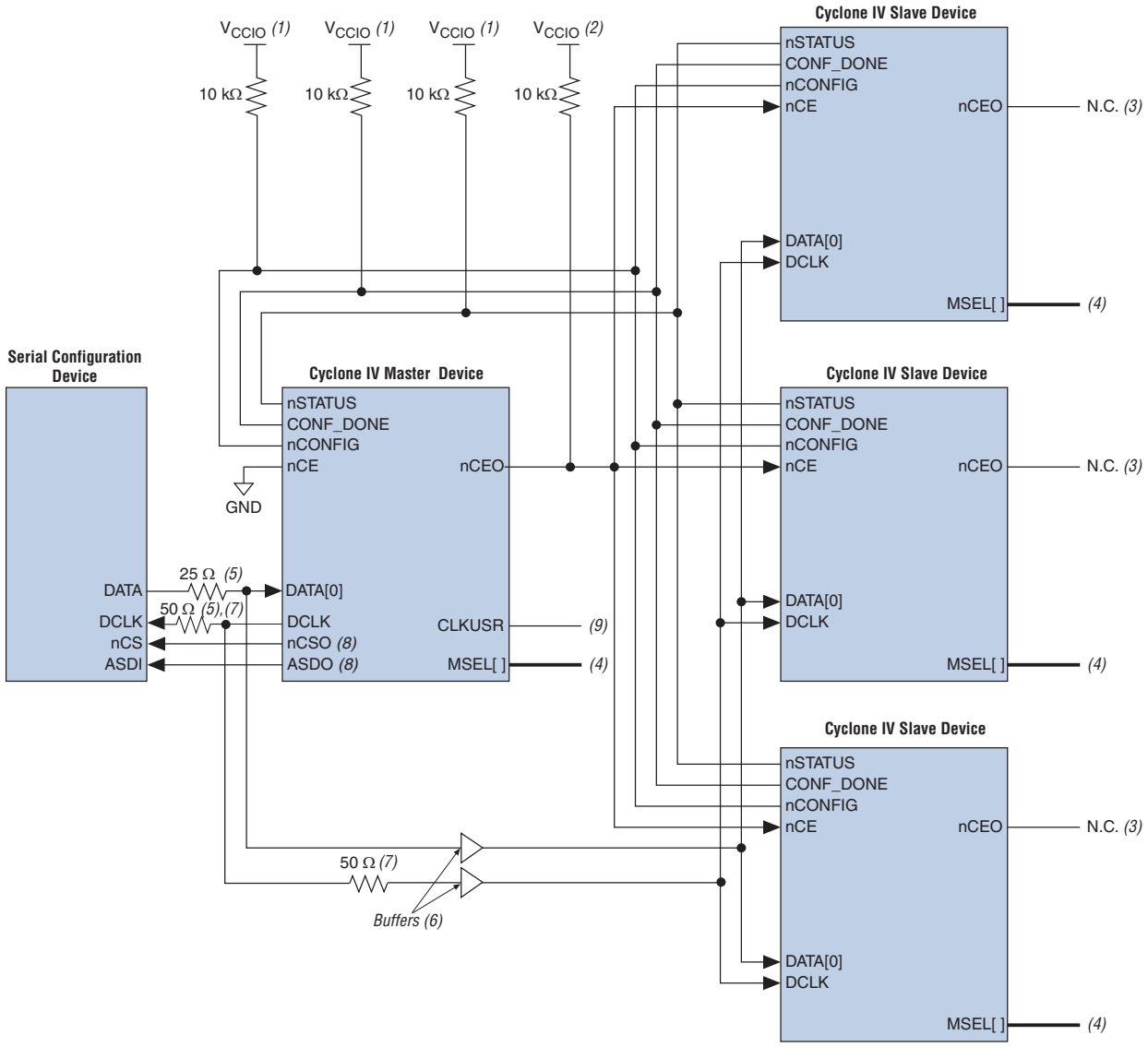
Two copies of the **.sof** are stored in the serial configuration device. Use the first copy to configure the master device of the Cyclone IV device and the second copy to configure all remaining slave devices concurrently. All slave devices must have the same density and package. The setup is similar to Figure 8–3 on page 8–13.

To configure four identical Cyclone IV devices with the same **.sof**, you must set up the chain similar to the example shown in Figure 8–4. The first device is the master device and its MSEL pins must be set to select AS configuration. The other three slave devices are set up for concurrent configuration and their MSEL pins must be set to select PS configuration. The nCEO pin from the master device drives the nCE input pins on all three slave devices, as well as the DATA and DCLK pins that connect in parallel to all four devices. During the first configuration cycle, the master device reads its configuration data from the serial configuration device while holding nCEO high. After completing its configuration cycle, the master device drives nCE low and sends the second copy of the configuration data to all three slave devices, configuring them simultaneously.

The advantage of the setup in Figure 8–4 is that you can have a different **.sof** for the master device. However, all the slave devices must be configured with the same **.sof**. You can either compress or uncompress the **.sof** in this configuration method.

☞ You can still use this method if the master and slave devices use the same **.sof**.

Figure 8–4. Multi-Device AS Configuration in Which Devices Receive the Same Data with Multiple .sof



Notes to Figure 8–4:

(1) Connect the pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) Connect the pull-up resistor to the V_{CCIO} supply voltage of the I/O bank in which the nCE pin resides.

(3) The nCEO pin is left unconnected or used as a user I/O pin when it does not feed the nCE pin of another device.

(4) The MSEL pin settings vary for different configuration voltage standards and POR time. You must set the master device in AS mode and the slave devices in PS mode. To connect the MSEL pins for the master device in AS mode and the slave devices in PS mode, refer to Table 8–3 on page 8–8, Table 8–4 on page 8–8, and Table 8–5 on page 8–9. Connect the MSEL pins directly to V_{CCA} or GND.

(5) Connect the series resistor at the near end of the serial configuration device.

(6) Connect the repeater buffers between the master and slave devices for DATA[0] and DCLK. All I/O inputs must maintain a maximum AC voltage of 4.1 V. The output resistance of the repeater buffers must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 8–5.

(7) The 50-Ω series resistors are optional if the 3.3-V configuration voltage standard is applied. For optimal signal integrity, connect these 50-Ω series resistors if the 2.5- or 3.0-V configuration voltage standard is applied.

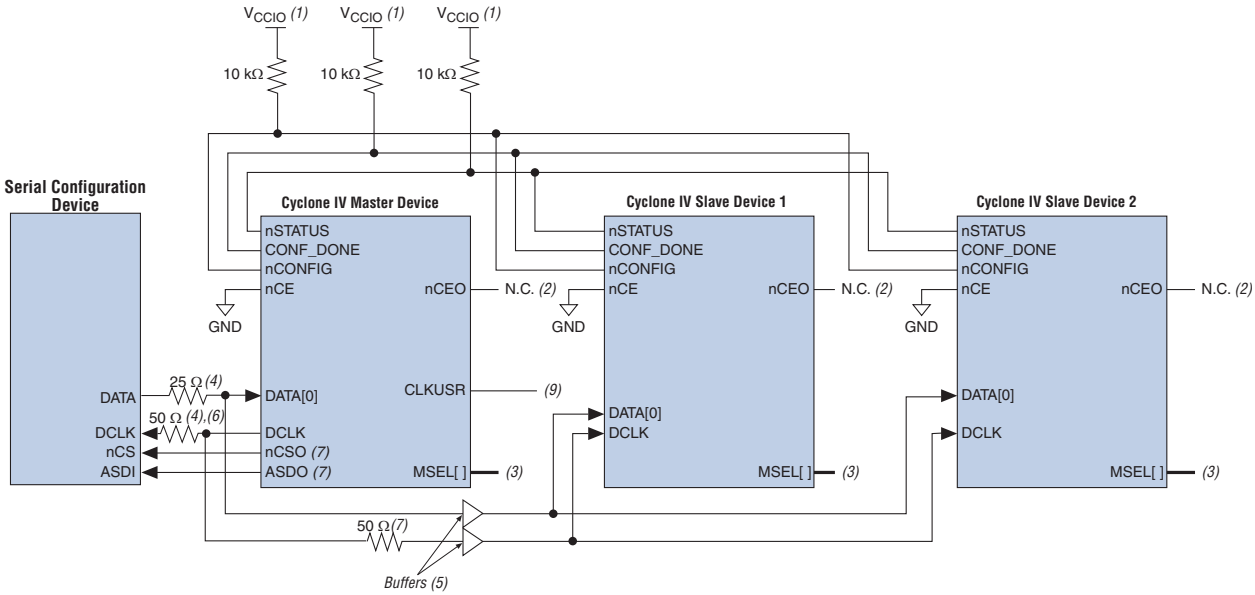
(8) These pins are dual-purpose I/O pins. The nCSO pin functions as FLASH_nCE pin in AP mode. The ASDO pin functions as DATA[1] pin in AP and FPP modes.

(9) Only Cyclone IV GX devices have an option to select CLKUSR (40 MHz maximum) as the external clock source for DCLK.

Single SRAM Object File

The second method configures both the master device and slave devices with the same **.sof**. The serial configuration device stores one copy of the **.sof**. You must set up one or more slave devices in the chain. All the slave devices must be set up in the same way (Figure 8–5).

Figure 8–5. Multi-Device AS Configuration in Which Devices Receive the Same Data with a Single .sof



Notes to Figure 8–5:

(1) Connect the pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) The nCEO pin is left unconnected or used as a user I/O pin when it does not feed the nCE pin of another device.

(3) The MSEL pin settings vary for different configuration voltage standards and POR time. You must set the master device of the Cyclone IV device in AS mode and the slave devices in PS mode. To connect the MSEL pins for the master device in AS mode and slave devices in PS mode, refer to Table 8–3 on page 8–8, Table 8–4 on page 8–8, and Table 8–5 on page 8–9. Connect the MSEL pins directly to V_{CCA} or GND.

(4) Connect the series resistor at the near end of the serial configuration device.

(5) Connect the repeater buffers between the master and slave devices for DATA[0] and DCLK. All I/O inputs must maintain a maximum AC voltage of 4.1 V. The output resistance of the repeater buffers must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 8–5.

(6) The 50-Ω series resistors are optional if the 3.3-V configuration voltage standard is applied. For optimal signal integrity, connect these 50-Ω series resistors if the 2.5- or 3.0-V configuration voltage standard is applied.

(7) These pins are dual-purpose I/O pins. The nCSO pin functions as FLASH_nCE pin in AP mode. The ASDO pin functions as DATA[1] pin in AP and FPP modes.

(8) Only Cyclone IV GX devices have an option to select CLKUSR (40 MHz maximum) as the external clock source for DCLK.

In this setup, all the Cyclone IV devices in the chain are connected for concurrent configuration. This reduces the AS configuration time because all the Cyclone IV devices are configured in one configuration cycle. Connect the nCE input pins of all the Cyclone IV devices to GND. You can either leave the nCEO output pins on all the Cyclone IV devices unconnected or use the nCEO output pins as normal user I/O pins. The DATA and DCLK pins are connected in parallel to all the Cyclone IV devices.

Altera recommends putting a buffer before the DATA and DCLK output from the master device to avoid signal strength and signal integrity issues. The buffer must not significantly change the DATA-to-DCLK relationships or delay them with respect to other AS signals (ASDI and nCS). Also, the buffer must only drive the slave devices to ensure that the timing between the master device and the serial configuration device is unaffected.

This configuration method supports both compressed and uncompressed **.sof**. Therefore, if the configuration bitstream size exceeds the capacity of a serial configuration device, you can enable the compression feature in the **.sof** or you can select a larger serial configuration device.

Guidelines for Connecting a Serial Configuration Device to Cyclone IV Devices for an AS Interface

For single- and multi-device AS configurations, the board trace length and loading between the supported serial configuration device and Cyclone IV device must follow the recommendations listed in Table 8–6.

Table 8–6. Maximum Trace Length and Loading for AS Configuration

Cyclone IV Device AS Pins	Maximum Board Trace Length from a Cyclone IV Device to a Serial Configuration Device (Inches)		Maximum Board Load (pF)
	Cyclone IV E	Cyclone IV GX	
DCLK	10	6	15
DATA[0]	10	6	30
nCSO	10	6	30
ASDO	10	6	30

Estimating AS Configuration Time

AS configuration time is dominated by the time it takes to transfer data from the serial configuration device to the Cyclone IV device. This serial interface is clocked by the Cyclone IV device DCLK output (generated from a 40-MHz internal oscillator for Cyclone IV E devices, a 20- or 40-MHz internal oscillator, or an external CLKUSR of up to 40 MHz for Cyclone IV GX devices). Equation 8–2 and Equation 8–3 show the configuration time calculations.

Equation 8–2.

$$\text{RBF Size} \times \left(\frac{\text{maximum DCLK period}}{\text{1 bit}} \right) = \text{estimated maximum configuration time}$$

Equation 8–3.

$$9{,}600{,}000 \text{ bits} \times \left(\frac{50 \text{ ns}}{\text{1 bit}} \right) = 480 \text{ ms}$$

Enabling compression reduces the amount of configuration data that is sent to the Cyclone IV device, which also reduces configuration time. On average, compression reduces configuration time by 50%.

Programming Serial Configuration Devices

Serial configuration devices are non-volatile, flash memory-based devices. You can program these devices in-system with the USB-Blaster™ or ByteBlaster™ II download cables. Alternatively, you can program them with the Altera Programming Unit (APU), supported third-party programmers, or a microprocessor with the SRunner software driver.

You can perform in-system programming of serial configuration devices through the AS programming interface. During in-system programming, the download cable disables device access to the AS interface by driving the nCE pin high. Cyclone IV devices are also held in reset by a low level on nCONFIG. After programming is complete, the download cable releases nCE and nCONFIG, allowing the pull-down and pull-up resistors to drive V_{CC} and GND, respectively.

To perform in-system programming of a serial configuration device through the AS programming interface, you must place the diodes and capacitors as close as possible to the Cyclone IV device. You must ensure that the diodes and capacitors maintain a maximum AC voltage of 4.1 V (Figure 8–6).

If you want to use the setup shown in Figure 8–6 to perform in-system programming of a serial configuration device and single- or multi-device AS configuration, you do not require a series resistor on the DATA line at the near end of the serial configuration device. The existing diodes and capacitors are sufficient.

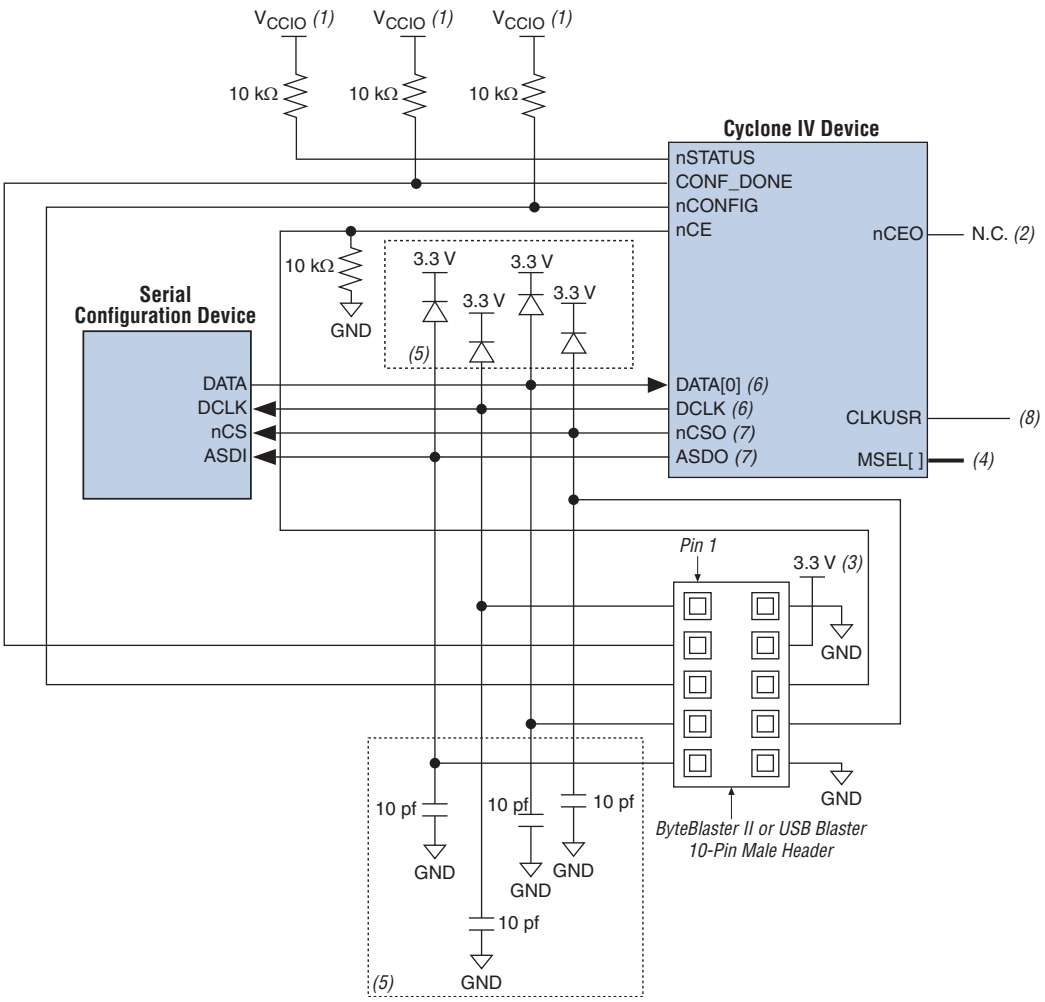
Altera has developed the Serial FlashLoader (SFL), a JTAG-based in-system programming solution for Altera serial configuration devices. The SFL is a bridge design for the Cyclone IV device that uses its JTAG interface to access the EPCS JIC (JTAG Indirect Configuration Device Programming) file and then uses the AS interface to program the EPCS device. Both the JTAG interface and AS interface are bridged together inside the SFL design.

For more information about implementing the SFL with Cyclone IV devices, refer to *AN 370: Using the Serial FlashLoader with the Quartus II Software*.

For more information about the USB-Blaster download cable, refer to the *USB-Blaster Download Cable User Guide*. For more information about the ByteBlaster II download cable, refer to the *ByteBlaster II Download Cable User Guide*.

Figure 8–6 shows the download cable connections to the serial configuration device.

Figure 8–6. In-System Programming of Serial Configuration Devices



Notes to Figure 8–6:

(1) Connect these pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) The nCEO pin is left unconnected or used as a user I/O pin when it does not feed the nCE pin of another device.

(3) Power up the V_{CC} of the ByteBlaster II or USB-Blaster download cable with the 3.3-V supply.

(4) The MSEL pin settings vary for different configuration voltage standards and POR time. To connect the MSEL pins, refer to Table 8–3 on page 8–8, Table 8–4 on page 8–8, and Table 8–5 on page 8–9. Connect the MSEL pins directly to V_{CCA} or GND.

(5) The diodes and capacitors must be placed as close as possible to the Cyclone IV device. You must ensure that the diodes and capacitors maintain a maximum AC voltage of 4.1 V. The external diodes and capacitors are required to prevent damage to the Cyclone IV device AS configuration input pins due to possible overshoot when programming the serial configuration device with a download cable. Altera recommends using the Schottky diode, which has a relatively lower forward diode voltage (VF) than the switching and Zener diodes, for effective voltage clamping.

(6) When cascading Cyclone IV devices in a multi-device AS configuration, connect the repeater buffers between the master and slave devices for DATA[0] and DCLK. All I/O inputs must maintain a maximum AC voltage of 4.1 V. The output resistance of the repeater buffers must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 8–5.

(7) These pins are dual-purpose I/O pins. The nCSO pin functions as FLASH_nCE pin in AP mode. The ASDO pin functions as DATA[1] pin in AP and FPP modes.

(8) Only Cyclone IV GX devices have an option to select CLKUSR (40 MHz maximum) as the external clock source for DCLK.

You can use the Quartus II software with the APU and the appropriate configuration device programming adapter to program serial configuration devices. All serial configuration devices are offered in an 8- or 16-pin small outline integrated circuit (SOIC) package.

In production environments, serial configuration devices are programmed using multiple methods. Altera programming hardware or other third-party programming hardware is used to program blank serial configuration devices before they are mounted onto PCBs. Alternatively, you can use an on-board microprocessor to program the serial configuration device in-system by porting the reference C-based SRunner software driver provided by Altera.

A serial configuration device is programmed in-system by an external microprocessor with the SRunner software driver. The SRunner software driver is a software driver developed for embedded serial configuration device programming, which is easily customized to fit in different embedded systems. The SRunner software driver is able to read a Raw Programming Data (**.rpd**) file and write to serial configuration devices. The serial configuration device programming time, using the SRunner software driver, is comparable to the programming time with the Quartus II software.

For more information about the SRunner software driver, refer to *AN 418: SRunner: An Embedded Solution for Serial Configuration Device Programming* and the source code at the Altera website.

AP Configuration (Supported Flash Memories)

The AP configuration scheme is only supported in Cyclone IV E devices. In the AP configuration scheme, Cyclone IV E devices are configured using commodity 16-bit parallel flash memory. These external non-volatile configuration devices are industry standard microprocessor flash memories. The flash memories provide a fast interface to access configuration data. The speed up in configuration time is mainly due to the 16-bit wide parallel data bus, which is used to retrieve data from the flash memory.

Some of the smaller Cyclone IV E devices or package options do not support the AP configuration scheme. Table 8–7 lists the supported AP configuration scheme for each Cyclone IV E devices.

Table 8–7. Supported AP Configuration Scheme for Cyclone IV E Devices

Device	Package Options						
	E144	M164	U256	F256	U484	F484	F780
EP4CE6	—	—	—	—	—	—	—
EP4CE10	—	—	—	—	—	—	—
EP4CE15	—	—	—	—	—	✓	—
EP4CE22	—	—	—	—	—	—	—
EP4CE30	—	—	—	—	—	✓	✓
EP4CE40	—	—	—	—	✓	✓	✓
EP4CE55	—	—	—	—	✓	✓	✓
EP4CE75	—	—	—	—	✓	✓	✓
EP4CE115	—	—	—	—	—	✓	✓

During device configuration, Cyclone IV E devices read configuration data using the parallel interface and configure their SRAM cells. This scheme is referred to as the AP configuration scheme because the device controls the configuration interface. This scheme contrasts with the FPP configuration scheme, where an external host controls the interface.

AP Configuration Supported Flash Memories

The AP configuration controller in Cyclone IV E devices is designed to interface with two industry-standard flash families—the Numonyx StrataFlash® Embedded Memory P30 flash family and the Numonyx StrataFlash Embedded Memory P33 flash family. Unlike serial configuration devices, both of the flash families supported in AP configuration scheme are designed to interface with microprocessors. By configuring from an industry standard microprocessor flash which allows access to the flash after entering user mode, the AP configuration scheme allows you to combine configuration data and user data (microprocessor boot code) on the same flash memory.

The Numonyx P30 flash family and the P33 flash family support a continuous synchronous burst read mode at 40 MHz DCLK frequency for reading data from the flash. Additionally, the Numonyx P30 and P33 flash families have identical pin-out and adopt similar protocols for data access.

Cyclone IV E devices use a 40-MHz oscillator for the AP configuration scheme. The oscillator is the same oscillator used in the Cyclone IV E AS configuration scheme.

Table 8–8 lists the supported families of the commodity parallel flash for the AP configuration scheme.

Table 8–8. Supported Commodity Flash for AP Configuration Scheme for Cyclone IV E Devices *(Note 1)*

Flash Memory Density	Numonyx P30 Flash Family *(2)*	Numonyx P33 Flash Family *(3)*
64 Mbit	✓	✓
128 Mbit	✓	✓
256 Mbit	✓	✓

Notes to Table 8–8:

(1) The AP configuration scheme only supports flash memory speed grades of 40 MHz and above.
(2) 3.3- , 3.0-, 2.5-, and 1.8-V I/O options are supported for the Numonyx P30 flash family.
(3) 3.3-, 3.0- and 2.5-V I/O options are supported for the Numonyx P33 flash family.

Configuring Cyclone IV E devices from the Numonyx P30 and P33 family 512-Mbit flash memory is possible, but you must properly drive the extra address and FLASH_nCE pins as required by these flash memories.

To check for supported speed grades and package options, refer to the respective flash datasheets.

The AP configuration scheme in Cyclone IV E devices supports flash speed grades of 40 MHz and above. However, AP configuration for all these speed grades must be capped at 40 MHz. The advantage of faster speed grades is realized when your design in the Cyclone IV E devices accesses flash memory in user mode.

For more information about the operation of the Numonyx StrataFlash Embedded Memory P30 and P33 flash memories, search for the keyword "P30" or "P33" on the Numonyx website (www.numonyx.com) to obtain the P30 or P33 family datasheet.

Single-Device AP Configuration

The following groups of interface pins are supported in Numonyx P30 and P33 flash memories:

- Control pins
- Address pins
- Data pins

The following control signals are from the supported parallel flash memories:

- CLK
- active-low reset (RST#)
- active-low chip enable (CE#)
- active-low output enable (OE#)
- active-low address valid (ADV#)
- active-low write enable (WE#)

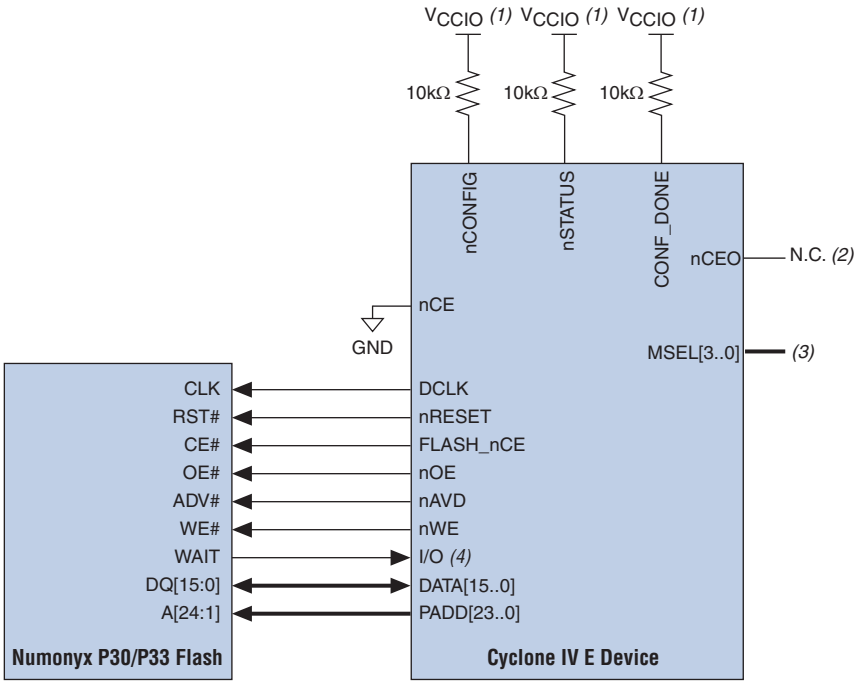
The supported parallel flash memories output a control signal (WAIT) to Cyclone IV E devices to indicate when synchronous data is ready on the data bus. Cyclone IV E devices have a 24-bit address bus connecting to the address bus (A[24:1]) of the flash memory. A 16-bit bidirectional data bus (DATA[15..0]) provides data transfer between the Cyclone IV E device and the flash memory.

The following control signals are from the Cyclone IV E device to flash memory:

- DCLK
- active-low hard rest (nRESET)
- active-low chip enable (FLASH_nCE)
- active-low output enable for the DATA[15..0] bus and WAIT pin (nOE)
- active-low address valid signal and is used to write data into the flash (nAVD)
- active-low write enable and is used to write data into the flash (nWE)

Figure 8–7 shows the interface for the Numonyx P30 flash memory and P33 flash memory to the Cyclone IV E device pins.

Figure 8–7. Single-Device AP Configuration Using Numonyx P30 and P33 Flash Memory



Notes to Figure 8–7:

(1) Connect the pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) The `nCEO` pin is left unconnected or used as a user I/O pin when it does not feed the `nCE` pin of another device.

(3) The `MSEL` pin settings vary for different configuration voltage standards and POR time. To connect `MSEL[3..0]`, refer to Table 8–5 on page 8–9. Connect the `MSEL` pins directly to V_{CCA} or GND.

(4) AP configuration ignores the `WAIT` signal during configuration mode. However, if you are accessing flash during user mode with user logic, you can optionally use normal I/O to monitor the `WAIT` signal from the Numonyx P30 or P33 flash.

In a single-device AP configuration, the maximum board loading and board trace length between supported parallel flash and Cyclone IV E devices must follow the recommendations listed in Table 8–9 on page 8–27.

If you use the AP configuration scheme for Cyclone IV E devices, the V_{CCIO} of I/O banks 1, 6, 7, and 8 must be 3.3, 3.0, 2.5, or 1.8 V. Altera does not recommend using the level shifter between the Numonyx P30 or P33 flash and the Cyclone IV E device in the AP configuration scheme.

There are no series resistors required in AP configuration mode for Cyclone IV E devices when using the Numonyx flash at 2.5-, 3.0-, and 3.3-V I/O standard. The output buffer of the Numonyx P30 IBIS model does not overshoot above 4.1 V. Thus, series resistors are not required for the 2.5-, 3.0-, and 3.3-V AP configuration option. However, if there are any other devices sharing the same flash I/Os with Cyclone IV E devices, all shared pins are still subject to the 4.1-V limit and may require series resistors.

Default read mode of the supported parallel flash memory and all writes to the parallel flash memory are asynchronous. Both the parallel flash families support a synchronous read mode, with data supplied on the positive edge of DCLK.

The serial clock (DCLK) generated by Cyclone IV E devices controls the entire configuration cycle and provides timing for the parallel interface.

Multi-Device AP Configuration

You can configure multiple Cyclone IV E devices using a single parallel flash. You can cascade multiple Cyclone IV E devices using the chip-enable (nCE) and chip-enable-out (nCEO) pins. The first device in the chain must have its nCE pin connected to GND. You must connect its nCEO pin to the nCE pin of the next device in the chain. Use an external 10-kΩ pull-up resistor to pull the nCEO signal high to its V_{CCIO} level to help the internal weak pull-up resistor. When the first device captures all its configuration data from the bitstream, it drives the nCEO pin low, enabling the next device in the chain. You can leave the nCEO pin of the last device unconnected or use it as a user I/O pin after configuration if the last device in the chain is a Cyclone IV E device. The nCONFIG, nSTATUS, CONF_DONE, DCLK, DATA[15..8], and DATA[7..0] pins of each device in the chain are connected (Figure 8–8 on page 8–25 and Figure 8–9 on page 8–26).

The first Cyclone IV E device in the chain, as shown in Figure 8–8 on page 8–25 and Figure 8–9 on page 8–26, is the configuration master device and controls the configuration of the entire chain. You must connect its MSEL pins to select the AP configuration scheme. The remaining Cyclone IV E devices are used as configuration slaves. You must connect their MSEL pins to select the FPP configuration scheme. Any other Altera device that supports FPP configuration can also be part of the chain as a configuration slave.

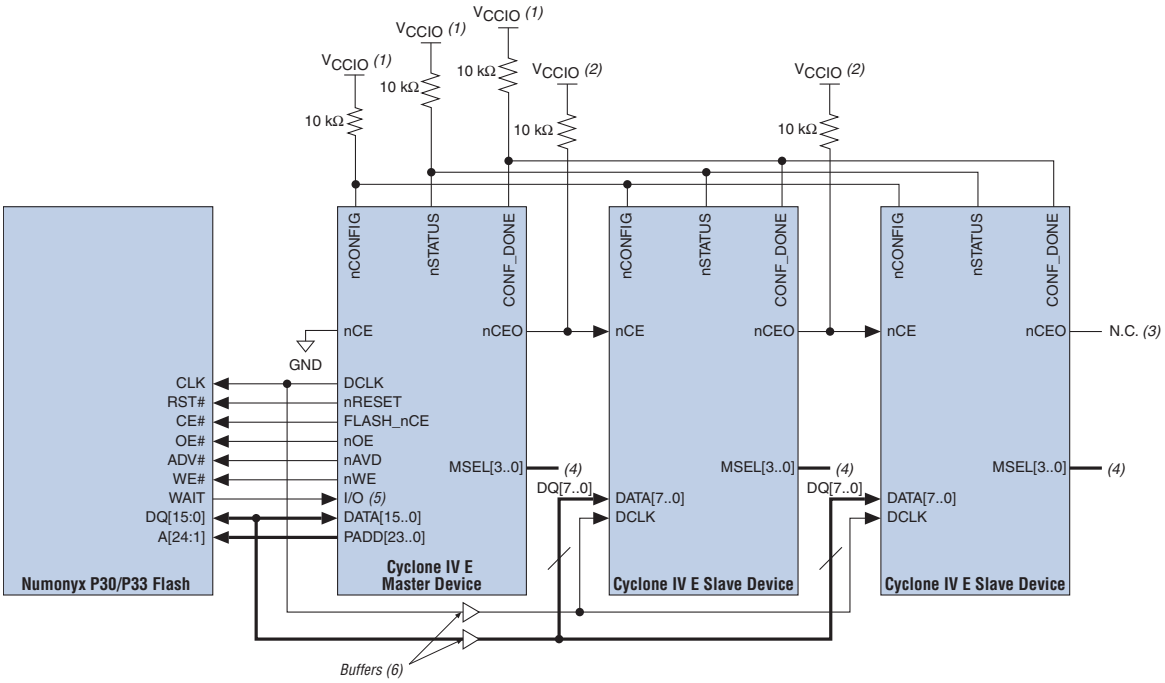
The following are the configurations for the DATA[15..0] bus in a multi-device AP configuration:

- Byte-wide multi-device AP configuration
- Word-wide multi-device AP configuration

Byte-Wide Multi-Device AP Configuration

The simpler method for multi-device AP configuration is the byte-wide multi-device AP configuration. In the byte-wide multi-device AP configuration, the LSB of the DATA[7..0] pin from the flash and master device (set to the AP configuration scheme) is connected to the slave devices set to the FPP configuration scheme, as shown in Figure 8–8.

Figure 8–8. Byte-Wide Multi-Device AP Configuration



Notes to Figure 8–8:

(1) Connect the pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) Connect the pull-up resistor to the V_{CCIO} supply voltage of the I/O bank in which the nCE pin resides.

(3) The nCEO pin is left unconnected or used as a user I/O pin when it does not feed the nCE pin of another device.

(4) The MSEL pin settings vary for different configuration voltage standards and POR time. You must set the master device in AP mode and the slave devices in FPP mode. To connect MSEL[3..0] for the master device in AP mode and the slave devices in FPP mode, refer to Table 8–5 on page 8–9. Connect the MSEL pins directly to V_{CCA} or GND.

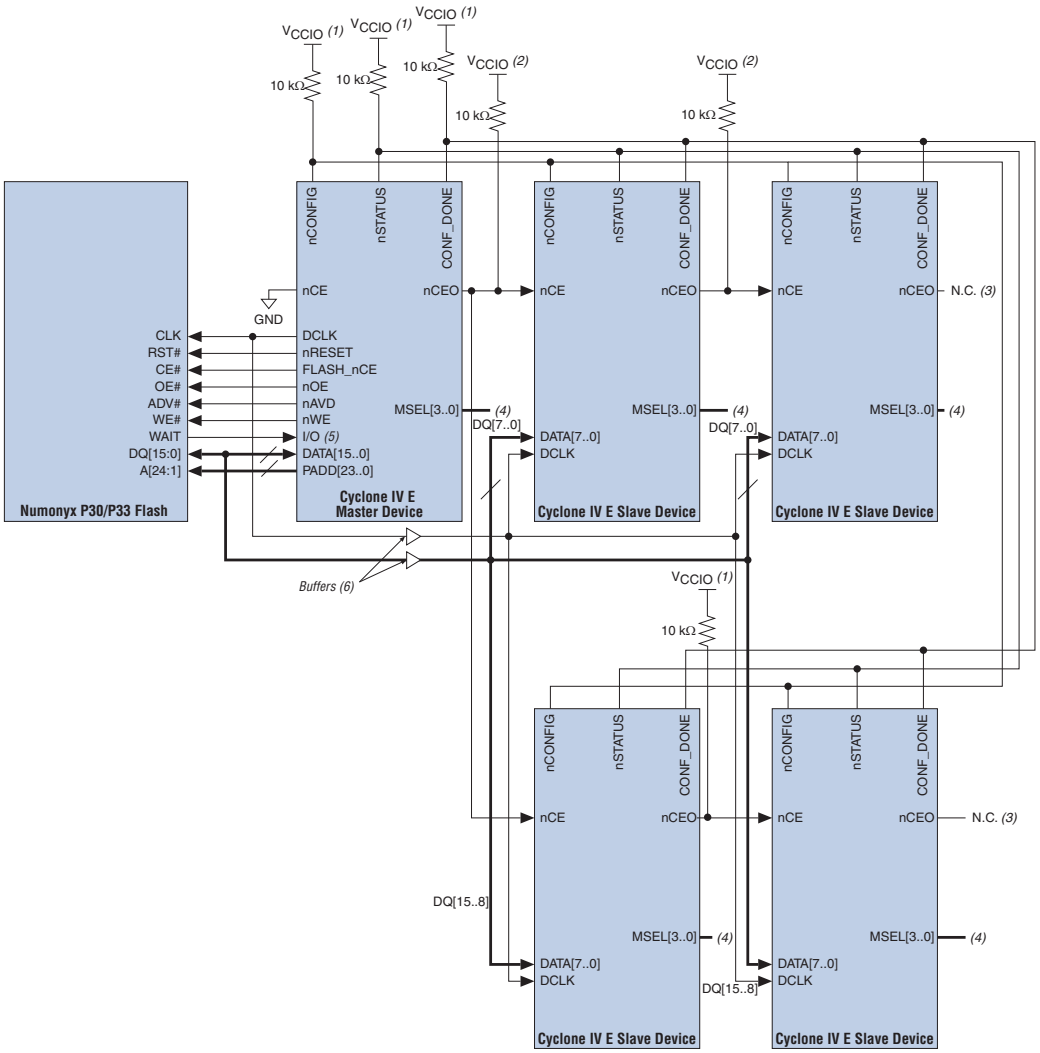
(5) The AP configuration ignores the WAIT signal during configuration mode. However, if you are accessing flash during user mode with user logic, you can optionally use the normal I/O to monitor the WAIT signal from the Numonyx P30 or P33 flash.

(6) Connect the repeater buffers between the Cyclone IV E master device and slave devices for DATA[15..0] and DCLK. All I/O inputs must maintain a maximum AC voltage of 4.1 V. The output resistance of the repeater buffers must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 8–5.

Word-Wide Multi-Device AP Configuration

The more efficient setup is one in which some of the slave devices are connected to the LSB of the DATA[7..0] and the remaining slave devices are connected to the MSB of the DATA[15..8]. In the word-wide multi-device AP configuration, the nCEO pin of the master device enables two separate daisy chains of slave devices, allowing both chains to be programmed concurrently, as shown in Figure 8–9.

Figure 8–9. Word-Wide Multi-Device AP Configuration



Notes to Figure 8–9:

(1) Connect the pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) Connect the pull-up resistor to the V_{CCIO} supply voltage of the I/O bank in which the nCE pin resides.

(3) The nCEO pin is left unconnected or used as a user I/O pin when it does not feed the nCE pin of another device.

(4) The MSEL pin settings vary for different configuration voltage standards and POR time. You must set the master device in AP mode and the slave devices in FPP mode. To connect MSEL[3..0] for the master device in AP mode and the slave devices in FPP mode, refer to Table 8–5 on page 8–9. Connect the MSEL pins directly to V_{CCA} or GND.

(5) The AP configuration ignores the WAIT signal during configuration mode. However, if you are accessing flash during user mode with user logic, you can optionally use the normal I/O pin to monitor the WAIT signal from the Numonyx P30 or P33 flash.

(6) Connect the repeater buffers between the Cyclone IV E master device and slave devices for DATA[15..0] and DCLK. All I/O inputs must maintain a maximum AC voltage of 4.1 V. The output resistance of the repeater buffers must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 8–5.

In a multi-device AP configuration, the board trace length between the parallel flash and the master device must follow the recommendations listed in Table 8–9.

The nSTATUS and CONF_DONE pins on all target devices are connected together with external pull-up resistors, as shown in Figure 8–8 on page 8–25 and Figure 8–9 on page 8–26. These pins are open-drain bidirectional pins on the devices. When the first device asserts nCEO (after receiving all its configuration data), it releases its CONF_DONE pin. However, the subsequent devices in the chain keep this shared CONF_DONE line low until they receive their configuration data. When all target devices in the chain receive their configuration data and release CONF_DONE, the pull-up resistor drives a high level on this line and all devices simultaneously enter initialization mode.

Guidelines for Connecting Parallel Flash to Cyclone IV E Devices for an AP Interface

For single- and multi-device AP configuration, the board trace length and loading between the supported parallel flash and Cyclone IV E devices must follow the recommendations listed in Table 8–9. These recommendations also apply to an AP configuration with multiple bus masters.

Table 8–9. Maximum Trace Length and Loading for AP Configuration

Cyclone IV E AP Pins	Maximum Board Trace Length from Cyclone IV E Device to Flash Device (inches)	Maximum Board Load (pF)
DCLK	6	15
DATA[15..0]	6	30
PADD[23..0]	6	30
nRESET	6	30
Flash_nCE	6	30
nOE	6	30
nAVD	6	30
nWE	6	30
I/O *(1)*	6	30

Note to Table 8–9:

(1) The AP configuration ignores the WAIT signal from the flash during configuration mode. However, if you are accessing flash during user mode with user logic, you can optionally use the normal I/O to monitor the WAIT signal from the Numonyx P30 or P33 flash.

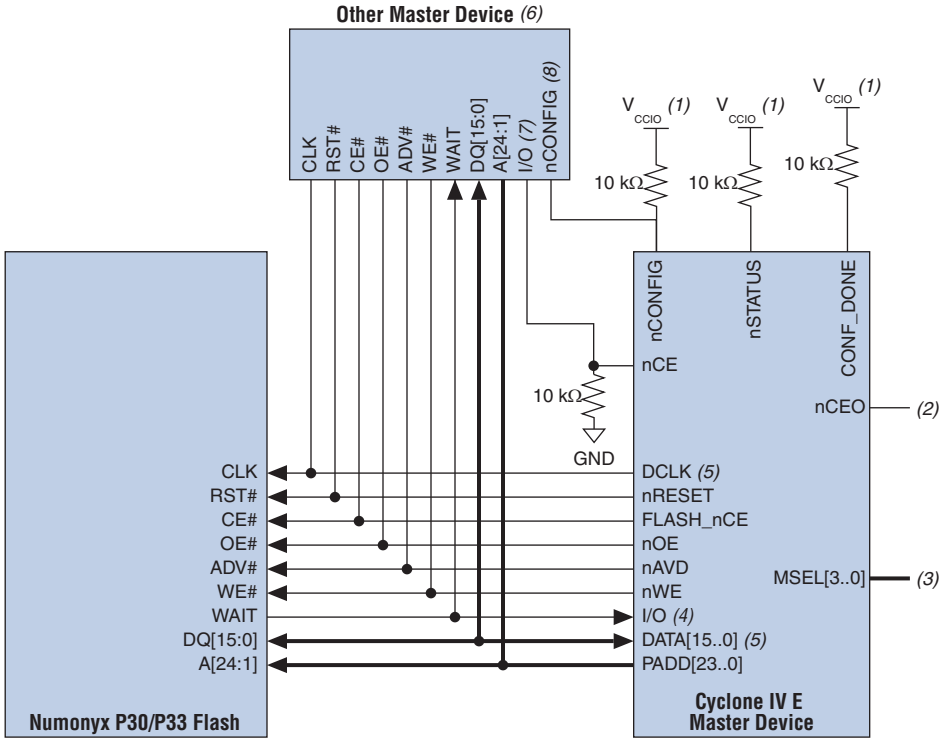
Configuring With Multiple Bus Masters

Similar to the AS configuration scheme, the AP configuration scheme supports multiple bus masters for the parallel flash. For another master to take control of the AP configuration bus, the master must assert nCONFIG low for at least 500 ns to reset the master Cyclone IV E device and override the weak 10-kΩ pull-down resistor on the nCE pin. This resets the master Cyclone IV E device and causes it to tri-state its AP configuration bus. The other master device then takes control of the AP configuration bus. After the other master device is done, it releases the AP configuration bus, then releases the nCE pin, and finally pulses nCONFIG low to restart the configuration.

In the AP configuration scheme, multiple masters share the parallel flash. Similar to the AS configuration scheme, the bus control is negotiated by the nCE pin.

Figure 8–10 shows the AP configuration with multiple bus masters.

Figure 8–10. AP Configuration with Multiple Bus Masters



Notes to Figure 8–10:

(1) Connect the pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) The `nCEO` pin is left unconnected or used as a user I/O pin when it does not feed the `nCE` pin of another device.

(3) The `MSEL` pin settings vary for different configuration voltage standards and POR time. To connect `MSEL[3..0]`, refer to Table 8–5 on page 8–9. Connect the `MSEL` pins directly to V_{CCA} or GND.

(4) The AP configuration ignores the `WAIT` signal during configuration mode. However, if you are accessing flash during user mode with user logic, you can optionally use the normal I/O to monitor the `WAIT` signal from the Numonyx P30 or P33 flash.

(5) When cascading Cyclone IV E devices in a multi-device AP configuration, connect the repeater buffers between the master device and slave devices for `DATA[15..0]` and `DCLK`. All I/O inputs must maintain a maximum AC voltage of 4.1 V. The output resistance of the repeater buffers must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 8–5.

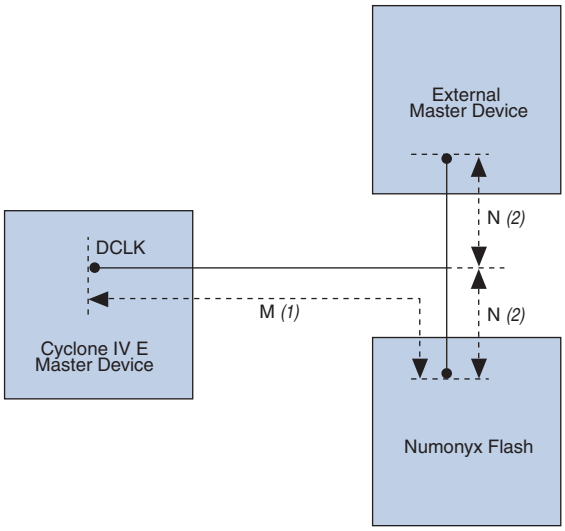
(6) The other master device must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 8–5.

(7) The other master device can control the AP configuration bus by driving the nCE to high with an output high on the I/O pin.

(8) The other master device can pulse `nCONFIG` if it is under system control and not tied to V_{CCIO}.

Figure 8–11 shows the recommended balanced star routing for multiple bus master interfaces to minimize signal integrity issues.

Figure 8–11. Balanced Star Routing



Notes to Figure 8–11:

(1) Altera recommends that *M* does not exceed 6 inches, as listed in Table 8–9 on page 8–27.

(2) Altera recommends using a balanced star routing. Keep the *N* length equal and as short as possible to minimize reflection noise from the transmission line. The *M* length is applicable for this setup.

Estimating AP Configuration Time

AP configuration time is dominated by the time it takes to transfer data from the parallel flash to Cyclone IV E devices. This parallel interface is clocked by the Cyclone IV E DCLK output (generated from an internal oscillator). The DCLK minimum frequency when using the 40-MHz oscillator is 20 MHz (50 ns). In word-wide cascade programming, the DATA[15..0] bus transfers a 16-bit word and essentially cuts configuration time to approximately 1/16 of the AS configuration time. Equation 8–4 and Equation 8–5 show the configuration time calculations.

Equation 8–4.

$$\text{RBF Size} \times \left(\frac{\text{maximum DCLK period}}{\text{16 bits per DCLK cycle}} \right) = \text{estimated maximum configuration time}$$

Equation 8–5.

$$9{,}600{,}000 \text{ bits} \times \left(\frac{50 \text{ ns}}{16 \text{ bit}} \right) = 30 \text{ ms}$$

Programming Parallel Flash Memories

Supported parallel flash memories are external non-volatile configuration devices. They are industry standard microprocessor flash memories. For more information about the supported families for the commodity parallel flash, refer to Table 8–8 on page 8–21.

Cyclone IV E devices in a single- or multiple-device chain support in-system programming of a parallel flash using the JTAG interface with the flash loader megafunction. The board intelligent host or download cable uses the four JTAG pins on Cyclone IV E devices to program the parallel flash in system, even if the host or download cable cannot access the configuration pins of the parallel flash.

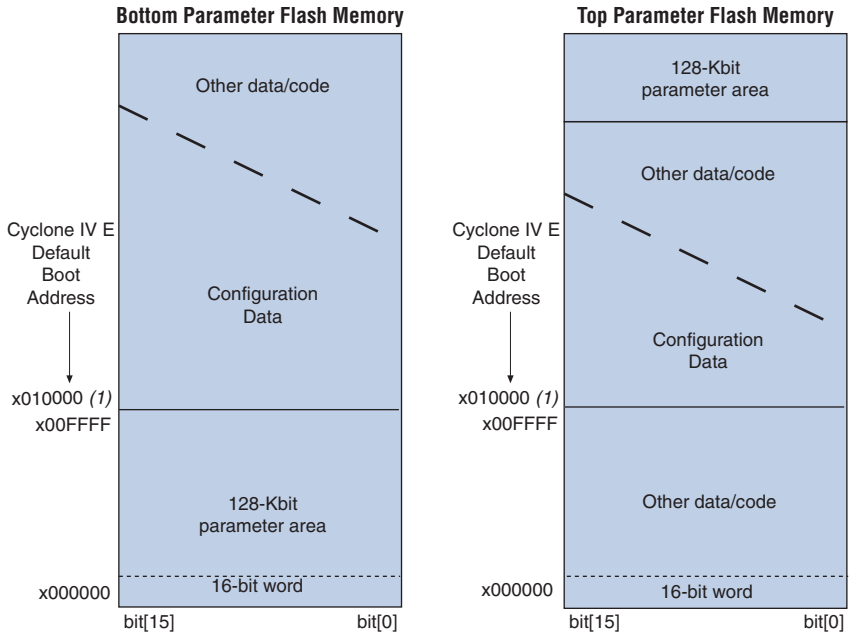
For more information about using the JTAG pins on Cyclone IV E devices to program the parallel flash in-system, refer to *AN 478: Using FPGA-Based Parallel Flash Loader (PFL) with the Quartus II Software*.

In the AP configuration scheme, the default configuration boot address is 0×010000 when represented in 16-bit word addressing in the supported parallel flash memory (Figure 8–12). In the Quartus II software, the default configuration boot address is 0×020000 because it is represented in 8-bit byte addressing. Cyclone IV E devices configure from word address 0×010000, which is equivalent to byte address 0×020000.

The Quartus II software uses byte addressing for the default configuration boot address. You must set the start address field to **0×020000**.

The default configuration boot address allows the system to use special parameter blocks in the flash memory map. Parameter blocks are at the top or bottom of the memory map. Figure 8–12 shows the configuration boot address in the AP configuration scheme. You can change the default configuration default boot address 0×010000 to any desired address using the `APFC_BOOT_ADDR` JTAG instruction. For more information about the `APFC_BOOT_ADDR` JTAG instruction, refer to "JTAG Instructions" on page 8–56.

Figure 8–12. Configuration Boot Address in AP Flash Memory Map



Note to Figure 8–12:

(1) The default configuration boot address is x010000 when represented in 16-bit word addressing.

PS Configuration

You can perform PS configuration on Cyclone IV devices with an external intelligent host, such as a MAX® II device, microprocessor with flash memory, or a download cable. In the PS scheme, an external host controls the configuration. Configuration data is clocked into the target Cyclone IV device through `DATA[0]` at each rising edge of `DCLK`.

If your system already contains a common flash interface (CFI) flash memory, you can use it for Cyclone IV device configuration storage as well. The MAX II PFL feature provides an efficient method to program CFI flash memory devices through the JTAG interface and the logic to control the configuration from the flash memory device to the Cyclone IV device.

For more information about the PFL, refer to *AN 386: Using the Parallel Flash Loader with the Quartus II Software*.

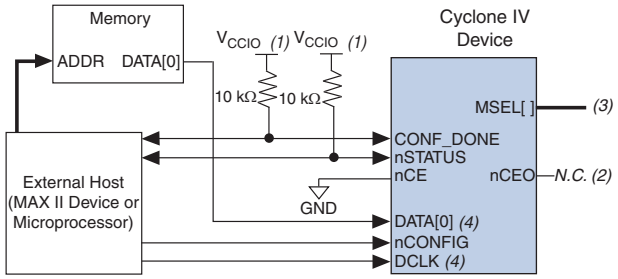
Cyclone IV devices do not support enhanced configuration devices for PS configuration.

PS Configuration Using an External Host

In the PS configuration scheme, you can use an intelligent host such as a MAX II device or microprocessor that controls the transfer of configuration data from a storage device, such as flash memory, to the target Cyclone IV device. You can store the configuration data in **.rbf**, **.hex**, or **.ttf** format.

Figure 8–13 shows the configuration interface connections between a Cyclone IV device and an external host device for single-device configuration.

Figure 8–13. Single-Device PS Configuration Using an External Host



Notes to Figure 8–13:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for the device. V_{CC} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

(2) The nCEO pin is left unconnected or used as a user I/O pin when it does not feed the nCE pin of another device.

(3) The MSEL pin settings vary for different configuration voltage standards and POR time. To connect the MSEL pins, refer to Table 8–3 on page 8–8, Table 8–4 on page 8–8, and Table 8–5 on page 8–9. Connect the MSEL pins directly to V_{CCA} or GND.

(4) All I/O inputs must maintain a maximum AC voltage of 4.1 V. DATA[0] and DCLK must fit the maximum overshoot outlined in Equation 8–1 on page 8–5.

To begin the configuration, the external host device must generate a low-to-high transition on the nCONFIG pin. When nSTATUS is pulled high, the external host device must place the configuration data one bit at a time on DATA[0]. If you use configuration data in **.rbf**, **.ttf**, or **.hex**, you must first send the LSB of each data byte. For example, if the **.rbf** contains the byte sequence 02 1B EE 01 FA, the serial bitstream you must send to the device is:

0100-0000 1101-1000 0111-0111 1000-0000 0101-1111

Cyclone IV devices receive configuration data on DATA[0] and the clock is received on DCLK. Data is latched into the device on the rising edge of DCLK. Data is continuously clocked into the target device until CONF_DONE goes high and the device enters initialization state.

Two DCLK falling edges are required after CONF_DONE goes high to begin the initialization of the device.

INIT_DONE is released and pulled high when initialization is complete. The external host device must be able to detect this low-to-high transition which signals the device has entered user mode. When initialization is complete, the device enters user mode. In user mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

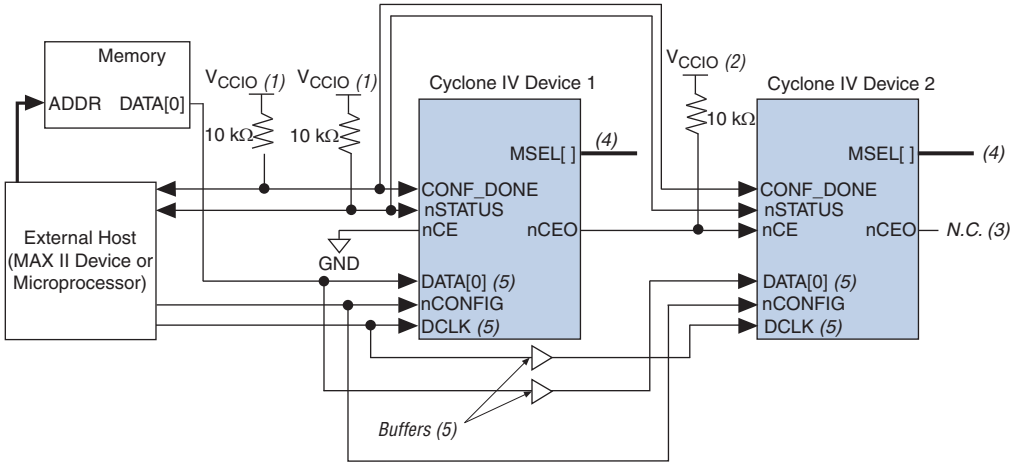
To ensure DCLK and DATA[0] are not left floating at the end of configuration, the MAX II device must drive them either high or low, whichever is convenient on your board. The DATA[0] pin is available as a user I/O pin after configuration. In the PS scheme, the DATA[0] pin is tri-stated by default in user mode and must be driven by the external host device. To change this default option in the Quartus II software, select the **Dual-Purpose Pins** tab of the **Device and Pin Options** dialog box.

The configuration clock (DCLK) speed must be below the specified system frequency to ensure correct configuration. No maximum DCLK period exists, which means you can pause configuration by halting DCLK for an indefinite amount of time.

The external host device can also monitor CONF_DONE and INIT_DONE to ensure successful configuration. The CONF_DONE pin must be monitored by the external device to detect errors and to determine when programming is complete. If all configuration data is sent, but CONF_DONE or INIT_DONE has not gone high, the external device must reconfigure the target device.

Figure 8–14 shows how to configure multiple devices using an external host device. This circuit is similar to the PS configuration circuit for a single device, except that Cyclone IV devices are cascaded for multi-device configuration.

Figure 8–14. Multi-Device PS Configuration Using an External Host



Notes to Figure 8–14:

(1) The pull-up resistor must be connected to a supply that provides an acceptable input signal for all devices in the chain. V_{CC} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

(2) Connect the pull-up resistor to the V_{CCIO} supply voltage of the I/O bank in which the nCE pin resides.

(3) The nCEO pin is left unconnected or used as a user I/O pin when it does not feed the nCE pin of another device.

(4) The MSEL pin settings vary for different configuration voltage standards and POR time. To connect the MSEL pins, refer to Table 8–3 on page 8–8, Table 8–4 on page 8–8, and Table 8–5 on page 8–9. Connect the MSEL pins directly to V_{CCA} or GND.

(5) All I/O inputs must maintain a maximum AC voltage of 4.1 V. DATA[0] and DCLK must fit the maximum overshoot outlined in Equation 8–1 on page 8–5.

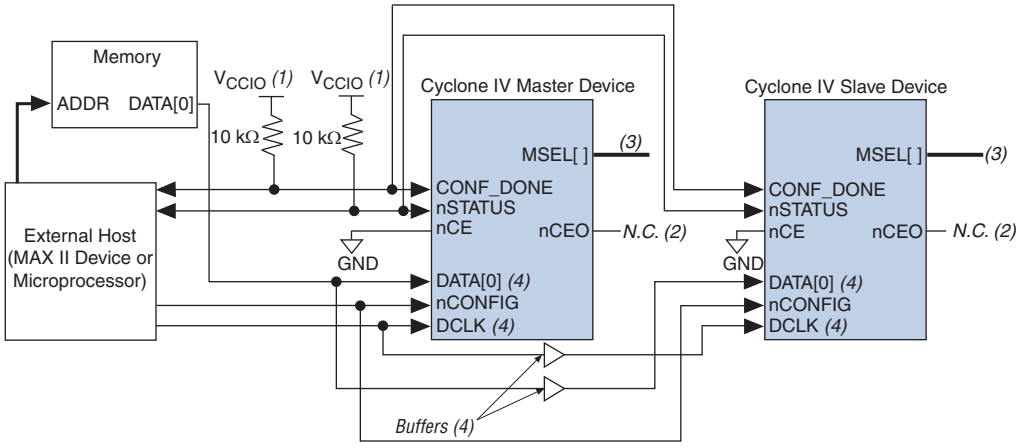
After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the nCE pin of the second device, which prompts the second device to begin configuration. The second device in the chain begins configuration in one clock cycle. Therefore, the transfer of data destinations is transparent to the external host device. nCONFIG, nSTATUS, DCLK, DATA[0], and CONF_DONE configuration pins are connected to every device in the chain. To ensure signal integrity and prevent clock skew problems, configuration signals may require buffering. Ensure that DCLK and DATA lines are buffered. All devices initialize and enter user mode at the same time because all CONF_DONE pins are tied together.

If any device detects an error, configuration stops for the entire chain and you must reconfigure the entire chain because all nSTATUS and CONF_DONE pins are tied together. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

You can have multiple devices that contain the same configuration data in your system. To support this configuration scheme, all device nCE inputs are tied to GND, while the nCEO pins are left floating. nCONFIG, nSTATUS, DCLK, DATA[0], and CONF_DONE configuration pins are connected to every device in the chain. To ensure signal integrity and prevent clock skew problems, configuration signals may require buffering. Ensure that the DCLK and DATA lines are buffered. Devices must be of the same density and package. All devices start and complete configuration at the same time.

Figure 8–15 shows a multi-device PS configuration when both Cyclone IV devices are receiving the same configuration data.

Figure 8–15. Multi-Device PS Configuration When Both Devices Receive the Same Data



Notes to Figure 8–15:

(1) You must connect the pull-up resistor to a supply that provides an acceptable input signal for all devices in the chain. V_{CC} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

(2) The nCEO pins of both devices are left unconnected or used as user I/O pins when configuring the same configuration data into multiple devices.

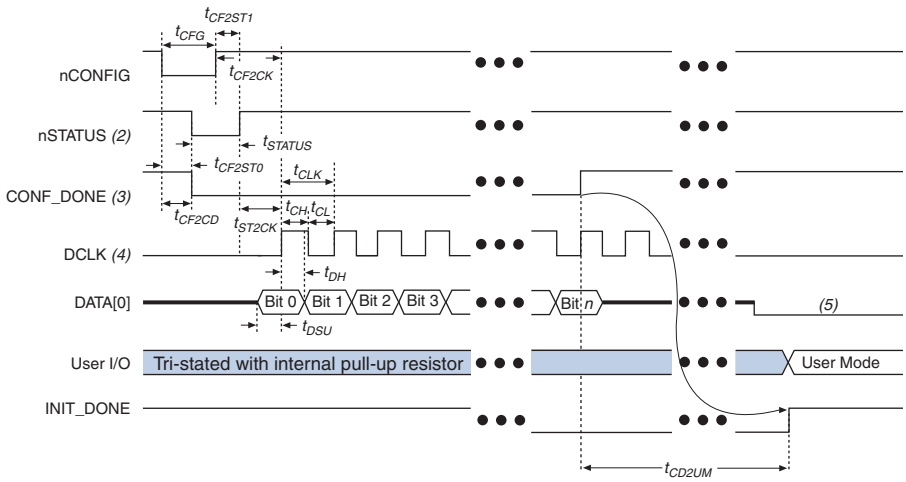
(3) The MSEL pin settings vary for different configuration voltage standards and POR time. To connect the MSEL pins, refer to Table 8–3 on page 8–8, Table 8–4 on page 8–8, and Table 8–5 on page 8–9. Connect the MSEL pins directly to V_{CCA} or GND.

(4) All I/O inputs must maintain a maximum AC voltage of 4.1 V. DATA[0] and DCLK must fit the maximum overshoot outlined in Equation 8–1 on page 8–5.

PS Configuration Timing

A PS configuration must meet the setup and hold timing parameters and the maximum clock frequency. When using a microprocessor or another intelligent host to control the PS interface, ensure that you meet these timing requirements. Figure 8–16 shows the timing waveform for PS configuration when using an external host device.

Figure 8–16. PS Configuration Timing Waveform *(Note 1)*



Notes to Figure 8–16:

(1) The beginning of this waveform shows the device in user mode. In user mode, `nCONFIG`, `nSTATUS`, and `CONF_DONE` are at logic-high levels. When `nCONFIG` is pulled low, a reconfiguration cycle begins.

(2) After power up, the Cyclone IV device holds `nSTATUS` low during POR delay.

(3) After power up, before and during configuration, `CONF_DONE` is low.

(4) In user mode, drive `DCLK` either high or low when using the PS configuration scheme, whichever is more convenient. When using the AS configuration scheme, `DCLK` is a Cyclone IV device output pin and must not be driven externally.

(5) Do not leave the `DATA[0]` pin floating after configuration. Drive the `DATA[0]` pin high or low, whichever is more convenient.

Table 8–10 lists the PS configuration timing parameters for Cyclone IV devices.

Table 8–10. PS Configuration Timing Parameters For Cyclone IV Devices (Part 1 of 2) *(Note 1)*

Symbol	Parameter	Minimum		Maximum		Unit
		Cyclone IV *(2)*	Cyclone IV E *(3)*	Cyclone IV *(2)*	Cyclone IV E *(3)*	
t_{CF2CD}	`nCONFIG` low to `CONF_DONE` low	—		500		ns
t_{CF2ST0}	`nCONFIG` low to `nSTATUS` low	—		500		ns
t_{CFG}	`nCONFIG` low pulse width	500		—		ns
t_{STATUS}	`nSTATUS` low pulse width	45		230 *(4)*		µs

Table 8–10. PS Configuration Timing Parameters For Cyclone IV Devices (Part 2 of 2) *(Note 1)*

Symbol	Parameter	Minimum		Maximum		Unit
		Cyclone IV *(2)*	Cyclone IV E *(3)*	Cyclone IV *(2)*	Cyclone IV E *(3)*	
t_{CF2ST1}	nCONFIG high to nSTATUS high	—		230 *(5)*		µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	230 *(4)*		—		µs
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2		—		µs
t_{DH}	Data hold time after rising edge on DCLK	0		—		ns
t_{CD2UM}	CONF_DONE high to user mode *(6)*	300		650		µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period		—		—
t_{CD2UMC}	CONF_DONE high to user mode with **CLKUSR** option on	t_{CD2CU} + (3,192 × CLKUSR period)		—		—
t_{DSU}	Data setup time before rising edge on DCLK	5	8	—	—	ns
t_{CH}	DCLK high time	3.2	6.4	—	—	ns
t_{CL}	DCLK low time	3.2	6.4	—	—	ns
t_{CLK}	DCLK period	7.5	15	—	—	ns
f_{MAX}	DCLK frequency *(7)*	—	—	133	66	MHz

Notes to Table 8–10:

(1) This information is preliminary.
(2) Applicable for Cyclone IV GX and Cyclone IV E devices with 1.2-V core voltage.
(3) Applicable for Cyclone IV E devices with 1.0-V core voltage.
(4) This value is applicable if you do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.
(5) This value is applicable if you do not delay configuration by externally holding the nSTATUS low.
(6) The minimum and maximum numbers apply only if you choose the internal oscillator as the clock source for starting the device.
(7) Cyclone IV E devices with 1.0-V core voltage have slower F_{MAX} when compared with Cyclone IV GX devices with 1.2-V core voltage.

PS Configuration Using a Download Cable

In this section, the generic term "download cable" includes the Altera USB-Blaster USB port download cable, MasterBlaster™ serial and USB communications cable, ByteBlaster II parallel port download cable, the ByteBlasterMV™ parallel port download cable, and the EthernetBlaster communications cable.

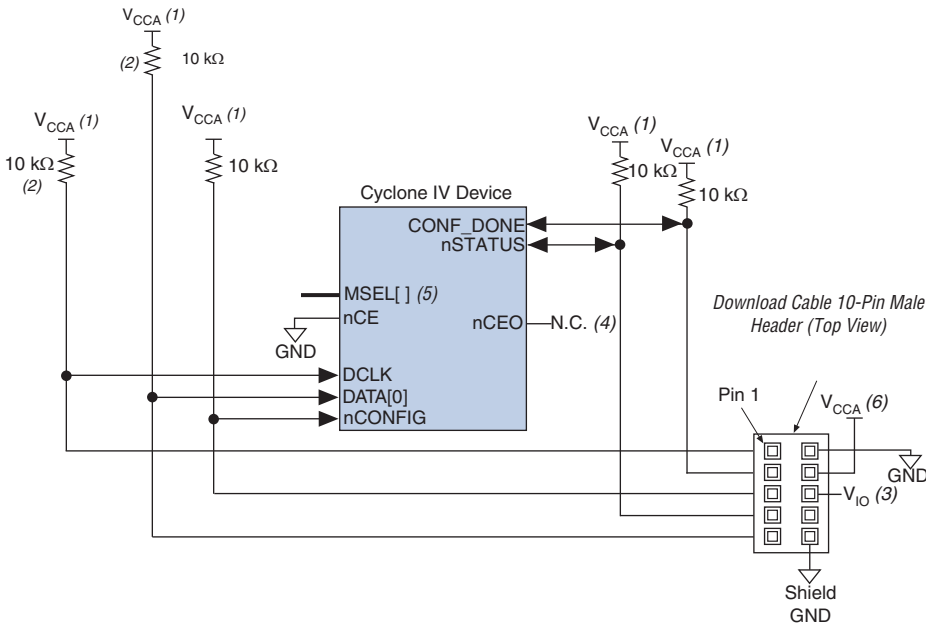
In the PS configuration with a download cable, an intelligent host (such as a PC) transfers data from a storage device to the Cyclone IV device through the download cable.

The programming hardware or download cable then places the configuration data one bit at a time on the DATA[0] pin of the device. The configuration data is clocked into the target device until CONF_DONE goes high. The CONF_DONE pin must have an external 10-kΩ pull-up resistor for the device to initialize.

When you use a download cable, setting the **Auto-restart configuration after error** option does not affect the configuration cycle because you must manually restart configuration in the Quartus II software if an error occurs. Additionally, the **Enable user-supplied start-up clock (CLKUSR)** option has no effect on device initialization, because this option is disabled in the **.sof** when programming the device with the Quartus II Programmer and download cable. Therefore, if you turn on the **CLKUSR** option, you do not have to provide a clock on CLKUSR when you configure the device with the Quartus II Programmer and a download cable.

Figure 8–17 shows PS configuration for Cyclone IV devices with a download cable.

Figure 8–17. PS Configuration Using a Download Cable



Notes to Figure 8–17:

(1) You must connect the pull-up resistor to the same supply voltage as the V_{CCA} supply.

(2) The pull-up resistors on DATA[0] and DCLK are only required if the download cable is the only configuration scheme used on your board. This is to ensure that DATA[0] and DCLK are not left floating after configuration. For example, if you also use a configuration device, the pull-up resistors on DATA[0] and DCLK are not required.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} must match the V_{CCA} of the device. For this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*. With the USB-Blaster, ByteBlaster II, ByteBlaster MV, and EthernetBlaster, this pin is a no connect.

(4) The nCEO pin is left unconnected or used as a user I/O pin when it does not feed the nCE pin of another device.

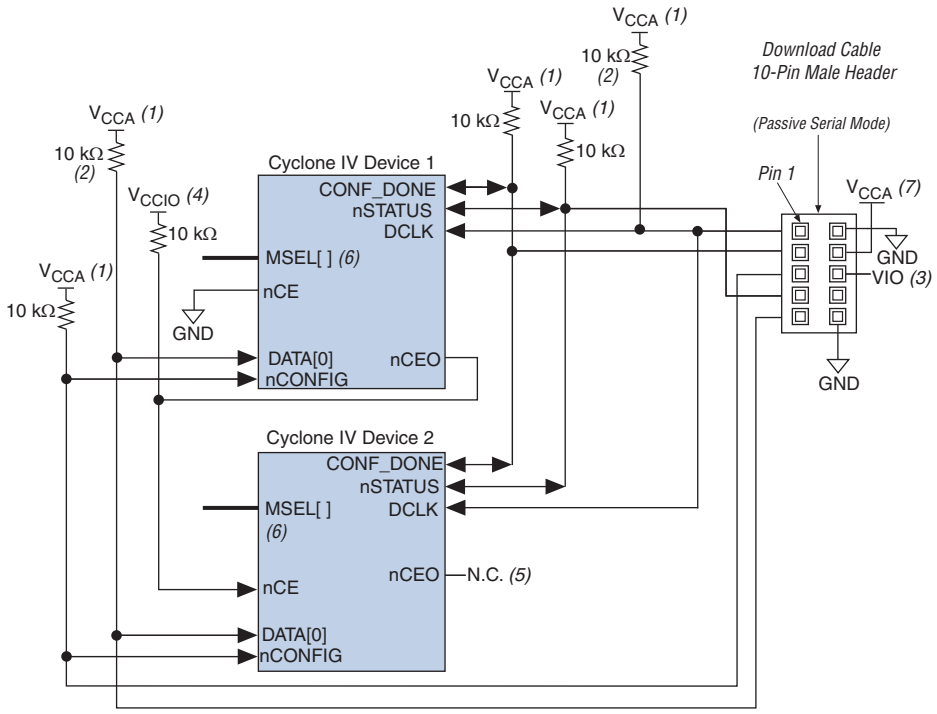
(5) The MSEL pin settings vary for different configuration voltage standards and POR time. To connect the MSEL pins, refer to Table 8–3 on page 8–8, Table 8–4 on page 8–8, and Table 8–5 on page 8–9 for PS configuration schemes. Connect the MSEL pins directly to V_{CCA} or GND.

(6) Power up the V_{CC} of the ByteBlaster II, USB-Blaster, or ByteBlasterMV cable with a 2.5-V supply from V_{CCA}. Third-party programmers must switch to 2.5 V. Pin 4 of the header is a V_{CC} power supply for the MasterBlaster cable. The MasterBlaster cable can receive power from either 5.0- or 3.3-V circuit boards, DC power supply, or 5.0 V from the USB cable. For this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*.

You can use a download cable to configure multiple Cyclone IV device configuration pins. nCONFIG, nSTATUS, DCLK, DATA[0], and CONF_DONE are connected to every device in the chain. All devices in the chain utilize and enter user mode at the same time because all CONF_DONE pins are tied together.

In addition, the entire chain halts configuration if any device detects an error because the nSTATUS pins are tied together. Figure 8–18 shows the PS configuration for multiple Cyclone IV devices using a MasterBlaster, USB-Blaster, ByteBlaster II, or ByteBlasterMV cable.

Figure 8–18. Multi-Device PS Configuration Using a Download Cable



Notes to Figure 8–18:

(1) You must connect the pull-up resistor to the same supply voltage as the V_{CCA} supply.

(2) The pull-up resistors on DATA[0] and DCLK are only required if the download cable is the only configuration scheme used on your board. This ensures that DATA[0] and DCLK are not left floating after configuration. For example, if you also use a configuration device, the pull-up resistors on DATA[0] and DCLK are not required.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} must match the V_{CCA} of the device. For this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*. When using the ByteBlasterMV download cable, this pin is a no connect. When using USB-Blaster, ByteBlaster II, and EthernetBlaster cables, this pin is connected to nCE when it is used for AS programming. Otherwise, it is a no connect.

(4) Connect the pull-up resistor to the V_{CCIO} supply voltage of the I/O bank in which the nCE pin resides.

(5) The nCEO pin of the last device in the chain is left unconnected or used as a user I/O pin.

(6) The MSEL pin settings vary for different configuration voltage standards and POR time. To connect MSEL for PS configuration schemes, refer to Table 8–3 on page 8–8, Table 8–4 on page 8–8, and Table 8–5 on page 8–9. Connect the MSEL pins directly to V_{CCA} or GND.

(7) Power up the V_{CC} of the ByteBlaster II, USB-Blaster, or ByteBlasterMV cable with a 2.5 V supply from V_{CCA}. Third-party programmers must switch to 2.5 V. Pin 4 of the header is a V_{CC} power supply for the MasterBlaster cable. The MasterBlaster cable can receive power from either 5.0- or 3.3-V circuit boards, DC power supply, or 5.0 V from the USB cable. For this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*.

FPP Configuration

The FPP configuration in Cyclone IV devices is designed to meet the increasing demand for faster configuration time. Cyclone IV devices are designed with the capability of receiving byte-wide configuration data per clock cycle.

You can perform FPP configuration of Cyclone IV devices with an intelligent host, such as a MAX II device or microprocessor with flash memory. If your system already contains a CFI flash memory, you can use it for the Cyclone IV device configuration storage as well. The MAX II PFL feature in MAX II devices provides an efficient method to program CFI flash memory devices through the JTAG interface and the logic to control configuration from the flash memory device to the Cyclone IV device.

For more information about the PFL, refer to *AN 386: Using the Parallel Flash Loader with the Quartus II Software*.

FPP configuration is supported in EP4CGX30 (only for F484 package), EP4CGX50, EP4CGX75, EP4CGX110, EP4CGX150, and all Cyclone IV E devices.

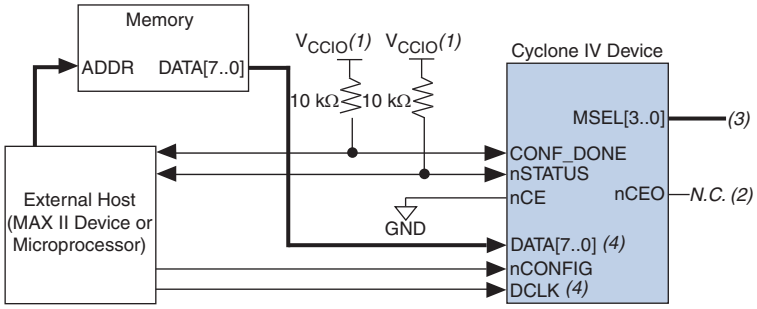
The FPP configuration is not supported in E144 package of Cyclone IV E devices.

Cyclone IV devices do not support enhanced configuration devices for FPP configuration.

FPP Configuration Using an External Host

FPP configuration using an external host provides a fast method to configure Cyclone IV devices. In the FPP configuration scheme, you can use an external host device to control the transfer of configuration data from a storage device, such as flash memory, to the target Cyclone IV device. You can store configuration data in an **.rbf**, **.hex**, or **.ttf** format. When using the external host, a design that controls the configuration process, such as fetching the data from flash memory and sending it to the device, must be stored in the external host device. Figure 8–19 shows the configuration interface connections between the Cyclone IV devices and an external device for single-device configuration.

Figure 8–19. Single-Device FPP Configuration Using an External Host



Notes to Figure 8–19:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for the device. V_{CC} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

(2) The nCEO pin is left unconnected or used as a user I/O pin when it does not feed the nCE pin of another device.

(3) The MSEL pin settings vary for different configuration voltage standards and POR time. To connect the MSEL pins, refer to Table 8–4 on page 8–8 and Table 8–5 on page 8–9. Connect the MSEL pins directly to V_{CCA} or GND.

(4) All I/O inputs must maintain a maximum AC voltage of 4.1 V. DATA[7..0] and DCLK must fit the maximum overshoot outlined in Equation 8–1 on page 8–5.

After nSTATUS is released, the device is ready to receive configuration data and the configuration stage begins. When nSTATUS is pulled high, the external host device places the configuration data one byte at a time on the DATA[7..0] pins.

Cyclone IV devices receive configuration data on the DATA[7..0] pins and the clock is received on the DCLK pin. Data is latched into the device on the rising edge of DCLK. Data is continuously clocked into the target device until CONF_DONE goes high. The CONF_DONE pin goes high one byte early in FPP configuration mode. The last byte is required for serial configuration (AS and PS) modes.

Two DCLK falling edges are required after CONF_DONE goes high to begin initialization of the device.

Supplying a clock on CLKUSR does not affect the configuration process. After the CONF_DONE pin goes high, CLKUSR is enabled after the time specified as t_{CD2CU}. After this time period elapses, Cyclone IV devices require 3,192 clock cycles to initialize properly and enter user mode. For more information about the supported CLKUSR f_{MAX} value for Cyclone IV devices, refer to Table 8–11 on page 8–43.

The INIT_DONE pin is released and pulled high when initialization is complete. The external host device must be able to detect this low-to-high transition, which signals the device has entered user mode. When initialization is complete, the device enters user mode. In user mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

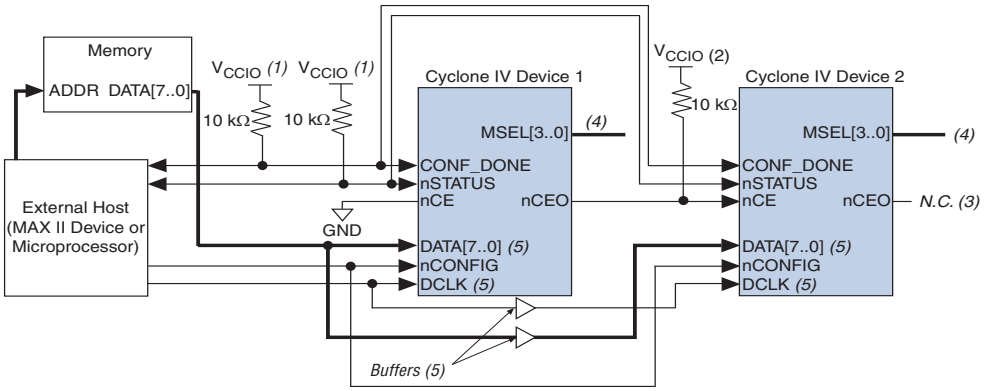
To ensure that DCLK and DATA[0] are not left floating at the end of the configuration, the MAX II device must drive them either high or low, whichever is convenient on your board. The DATA[0] pin is available as a user I/O pin after configuration. When you choose the FPP scheme in the Quartus II software, the DATA[0] pin is tri-stated by default in user mode and must be driven by the external host device. To change this default option in the Quartus II software, select the **Dual-Purpose Pins** tab of the **Device and Pin Options** dialog box.

The DCLK speed must be below the specified system frequency to ensure correct configuration. No maximum DCLK period exists, which means you can pause configuration by halting DCLK for an indefinite amount of time.

The external host device can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The CONF_DONE pin must be monitored by the external device to detect errors and to determine when programming is complete. If all configuration data is sent, but CONF_DONE or INIT_DONE has not gone high, the external device must reconfigure the target device.

Figure 8–20 shows how to configure multiple devices with a MAX II device. This circuit is similar to the FPP configuration circuit for a single device, except the Cyclone IV devices are cascaded for multi-device configuration.

Figure 8–20. Multi-Device FPP Configuration Using an External Host



Notes to Figure 8–20:

(1) The pull-up resistor must be connected to a supply that provides an acceptable input signal for all devices in the chain. V_{CC} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

(2) Connect the pull-up resistor to the V_{CCIO} supply voltage of the I/O bank in which the nCE pin resides.

(3) The nCEO pin is left unconnected or used as a user I/O pin when it does not feed the nCE pin of another device.

(4) The MSEL pin settings vary for different configuration voltage standards and POR time. To connect the MSEL pins, refer to Table 8–4 on page 8–8 and Table 8–5 on page 8–9. Connect the MSEL pins directly to V_{CCA} or GND.

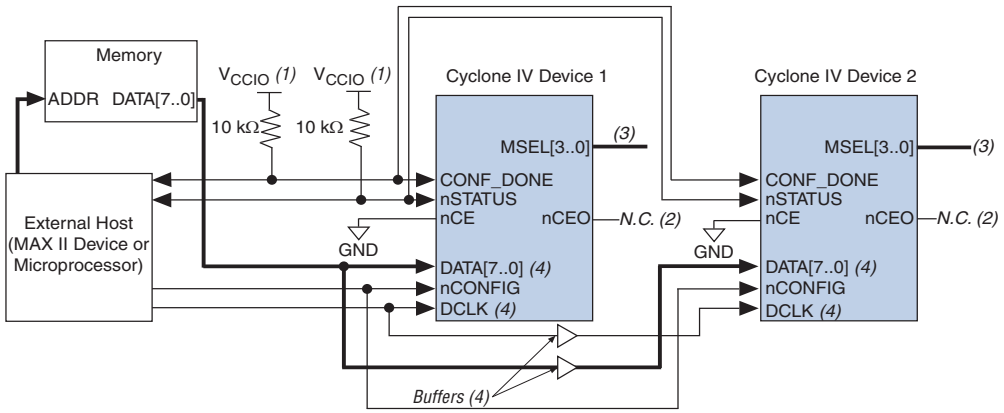
(5) All I/O inputs must maintain a maximum AC voltage of 4.1 V. DATA[7..0] and DCLK must fit the maximum overshoot outlined in Equation 8–1 on page 8–5.

After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the nCE pin of the second device, which prompts the second device to begin configuration. The second device in the chain begins configuration in one clock cycle; therefore, the transfer of data destinations is transparent to the MAX II device. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. Configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered. All devices initialize and enter user mode at the same time, because all device CONF_DONE pins are tied together.

All nSTATUS and CONF_DONE pins are tied together and if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

Figure 8–21 shows multi-device FPP configuration when both Cyclone IV devices are receiving the same configuration data. Configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. Configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered. Devices must be of the same density and package. All devices start and complete configuration at the same time.

Figure 8–21. Multi-Device FPP Configuration Using an External Host When Both Devices Receive the Same Data



Notes to Figure 8–21:

(1) You must connect the pull-up resistor to a supply that provides an acceptable input signal for all devices in the chain. V_{CC} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

(2) The nCEO pins of both devices are left unconnected or used as user I/O pins when configuring the same configuration data into multiple devices.

(3) The MSEL pin settings vary for different configuration voltage standards and POR time. To connect the MSEL pins, refer to Table 8–4 on page 8–8 and Table 8–5 on page 8–9. Connect the MSEL pins directly to V_{CCA} or GND.

(4) All I/O inputs must maintain a maximum AC voltage of 4.1 V. DATA[7..0] and DCLK must fit the maximum overshoot outlined in Equation 8–1 on page 8–5.

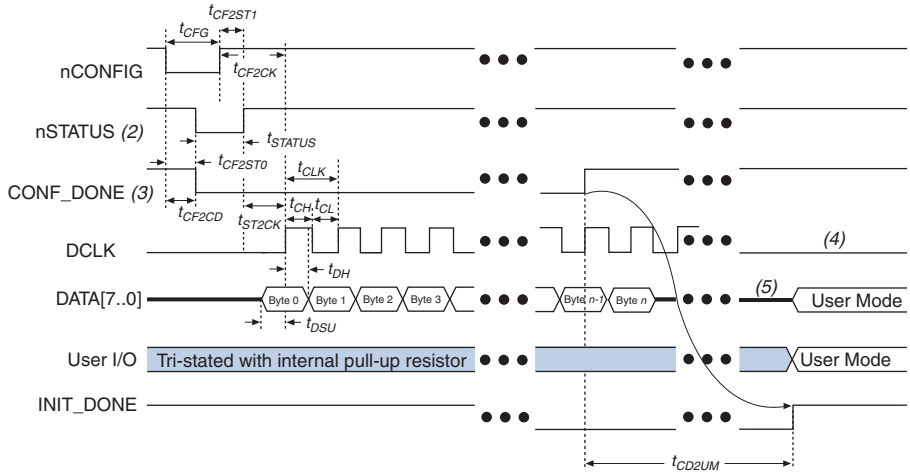
You can use a single configuration chain to configure Cyclone IV devices with other Altera devices that support FPP configuration. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device starts reconfiguration in all devices, tie all the CONF_DONE and nSTATUS pins together.

For more information about configuring multiple Altera devices in the same configuration chain, refer to *Configuring Mixed Altera FPGA Chains* in volume 2 of the *Configuration Handbook*.

FPP Configuration Timing

Figure 8–22 shows the timing waveform for the FPP configuration when using an external host.

Figure 8–22. FPP Configuration Timing Waveform *(Note 1)*



Notes to Figure 8–22:

(1) The beginning of this waveform shows the device in user mode. In user mode, nCONFIG, nSTATUS, and CONF_DONE are at logic-high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.
(2) After power up, the Cyclone IV device holds nSTATUS low during POR delay.
(3) After power up, before and during configuration, CONF_DONE is low.
(4) Do not leave DCLK floating after configuration. It must be driven high or low, whichever is more convenient.
(5) DATA[7..0] is available as a user I/O pin after configuration; the state of the pin depends on the dual-purpose pin settings.

Table 8–11 lists the FPP configuration timing parameters for Cyclone IV devices.

Table 8–11. FPP Timing Parameters for Cyclone IV Devices (Part 1 of 2) *(Note 1)*

Symbol	Parameter	Minimum		Maximum		Unit
		Cyclone IV *(2)*	Cyclone IV E *(3)*	Cyclone IV *(2)*	Cyclone IV E *(3)*	
t_{CF2CD}	nCONFIG low to CONF_DONE low	—		500		ns
t_{CF2ST0}	nCONFIG low to nSTATUS low	—		500		ns
t_{CFG}	nCONFIG low pulse width	500		—		ns
t_{STATUS}	nSTATUS low pulse width	45		230 *(4)*		µs
t_{CF2ST1}	nCONFIG high to nSTATUS high	—		230 *(5)*		µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	230 *(4)*		—		µs

Table 8–11. FPP Timing Parameters for Cyclone IV Devices (Part 2 of 2) *(Note 1)*

Symbol	Parameter	Minimum		Maximum		Unit
		Cyclone IV *(2)*	Cyclone IV E *(3)*	Cyclone IV *(2)*	Cyclone IV E *(3)*	
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2		—		µs
t_{DH}	Data hold time after rising edge on DCLK	0		—		ns
t_{CD2UM}	CONF_DONE high to user mode *(6)*	300		650		µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period		—		—
t_{CD2UMC}	CONF_DONE high to user mode with **CLKUSR** option on	t_{CD2CU} + (3,192 × CLKUSR period)		—		—
t_{DSU}	Data setup time before rising edge on DCLK	5	8	—	—	ns
t_{CH}	DCLK high time	3.2	6.4	—	—	ns
t_{CL}	DCLK low time	3.2	6.4	—	—	ns
t_{CLK}	DCLK period	7.5	15	—	—	ns
f_{MAX}	DCLK frequency *(7)*	—	—	133	66	MHz

Notes to Table 8–11:

(1) This information is preliminary.
(2) Applicable for Cyclone IV GX and Cyclone IV E with 1.2-V core voltage.
(3) Applicable for Cyclone IV E with 1.0-V core voltage.
(4) This value is applicable if you do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.
(5) This value is applicable if you do not delay configuration by externally holding the nSTATUS low.
(6) The minimum and maximum numbers apply only if you choose the internal oscillator as the clock source for starting the device.
(7) Cyclone IV E devices with 1.0-V core voltage have slower F_{MAX} when compared with Cyclone IV GX devices with 1.2-V core voltage.

JTAG Configuration

JTAG has developed a specification for boundary-scan testing (BST). The BST architecture offers the capability to efficiently test components on PCBs with tight lead spacing. The BST architecture can test pin connections without using physical test probes and capture functional data while a device is normally operating. You can also use the JTAG circuitry to shift configuration data into the device. The Quartus II software automatically generates **.sof** for JTAG configuration with a download cable in the Quartus II software Programmer.

For more information about the JTAG boundary-scan testing, refer to the *JTAG Boundary-Scan Testing for Cyclone IV Devices* chapter.

JTAG instructions have precedence over any other configuration modes. Therefore, JTAG configuration can take place without waiting for other configuration modes to complete. For example, if you attempt JTAG configuration in Cyclone IV devices during PS configuration, PS configuration terminates and JTAG configuration begins. If the MSEL pins are set to AS mode, the Cyclone IV device does not output a DCLK signal when JTAG configuration takes place.

The four required pins for a device operating in JTAG mode are TDI, TDO, TMS, and TCK. All the JTAG input pins are powered by the V_{CCIO} pin and support the LVTTL I/O standard only. All user I/O pins are tri-stated during JTAG configuration. Table 8–12 explains the function of each JTAG pin.

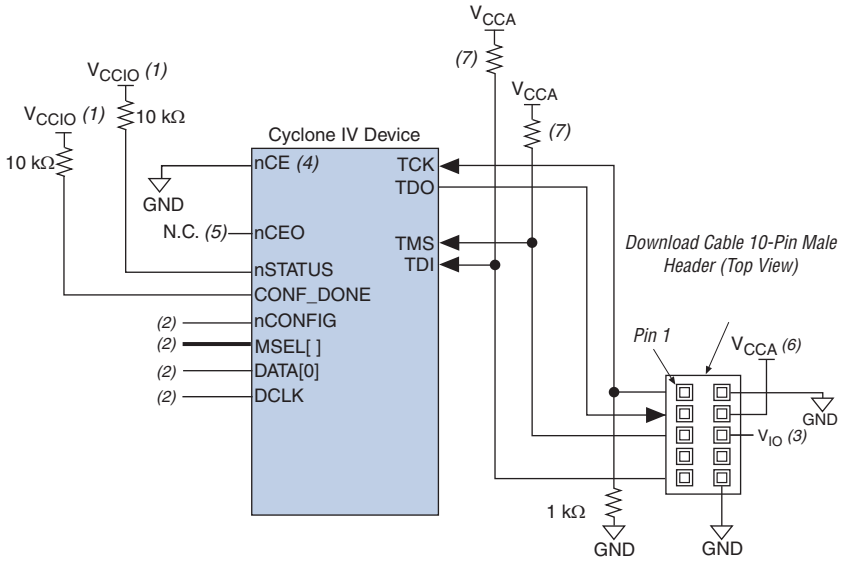
Table 8–12. Dedicated JTAG Pins

Pin Name	Pin Type	Description
TDI	Test data input	Serial input pin for instructions as well as test and programming data. Data shifts in on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry is disabled by connecting this pin to V_{CC}. TDI pin has weak internal pull-up resistors (typically 25 kΩ).
TDO	Test data output	Serial data output pin for instructions as well as test and programming data. Data shifts out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. If the JTAG interface is not required on the board, the JTAG circuitry is disabled by leaving this pin unconnected.
TMS	Test mode select	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions in the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry is disabled by connecting this pin to V_{CC}. TMS pin has weak internal pull-up resistors (typically 25 kΩ).
TCK	Test clock input	The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. If the JTAG interface is not required on the board, the JTAG circuitry is disabled by connecting this pin to GND. The TCK pin has an internal weak pull-down resistor.

You can download data to the device through the USB-Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV download cable, or the EthernetBlaster communications cable during JTAG configuration. Configuring devices with a cable is similar to programming devices in-system. Figure 8–23 and Figure 8–24 show the JTAG configuration of a single Cyclone IV device.

For device using V_{CCIO} of 2.5, 3.0, and 3.3 V, refer to Figure 8–23. All I/O inputs must maintain a maximum AC voltage of 4.1 V because JTAG pins do not have the internal PCI clamping diodes to prevent voltage overshoot when using V_{CCIO} of 2.5, 3.0, and 3.3 V. You must power up the V_{CC} of the download cable with a 2.5-V supply from V_{CCA}. For device using V_{CCIO} of 1.2, 1.5, and 1.8 V, refer to Figure 8–24. You can power up the V_{CC} of the download cable with the supply from V_{CCIO}.

Figure 8–23. JTAG Configuration of a Single Device Using a Download Cable (2.5, 3.0, and 3.3-V V_{CCIO} Powering the JTAG Pins)



Notes to Figure 8–23:

(1) Connect these pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) Connect the `nCONFIG` and `MSEL` pins to support a non-JTAG configuration scheme. If you only use JTAG configuration, connect the `nCONFIG` pin to logic-high and the `MSEL` pins to GND. In addition, pull `DCLK` and `DATA[0]` to either high or low, whichever is convenient on your board.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} must match the device's V_{CCA}. For this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*. When using the USB-Blaster, ByteBlaster II, ByteBlasterMV, and EthernetBlaster cables, this pin is a no connect.

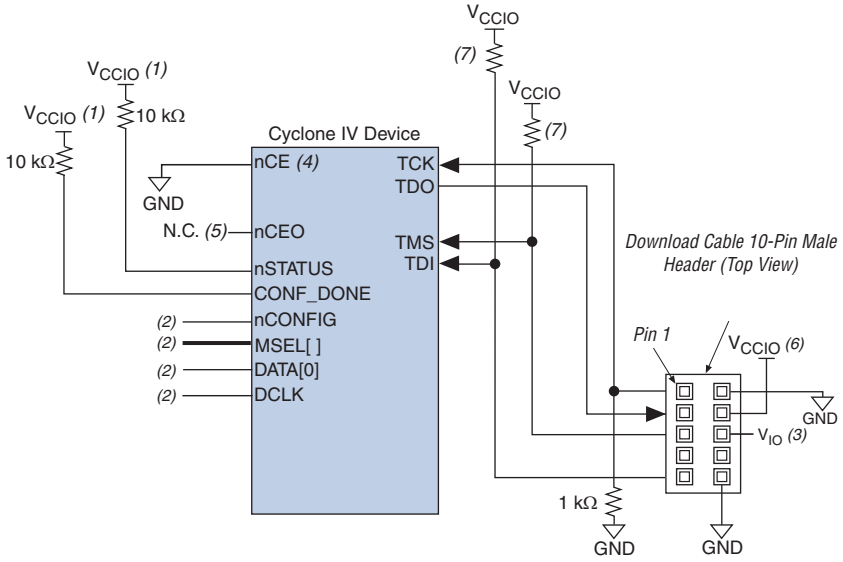
(4) The `nCE` pin must be connected to GND or driven low for successful JTAG configuration.

(5) The `nCEO` pin is left unconnected or used as a user I/O pin when it does not feed the `nCE` pin of another device.

(6) Power up the V_{CC} of the EthernetBlaster, ByteBlaster II, USB-Blaster, or ByteBlasterMV cable with a 2.5-V supply from V_{CCA}. Third-party programmers must switch to 2.5 V. Pin 4 of the header is a V_{CC} power supply for the MasterBlaster cable. The MasterBlaster cable can receive power from either 5.0- or 3.3-V circuit boards, DC power supply, or 5.0 V from the USB cable. For this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*.

(7) Resistor value can vary from 1 kΩ to 10 kΩ..

Figure 8–24. JTAG Configuration of a Single Device Using a Download Cable (1.5-V or 1.8-V V_{CCIO} Powering the JTAG Pins)



Notes to Figure 8–24:

(1) Connect these pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) Connect the nCONFIG and MSEL pins to support a non-JTAG configuration scheme. If you only use JTAG configuration, connect the nCONFIG pin to logic-high and the MSEL pins to GND. In addition, pull DCLK and DATA[0] to either high or low, whichever is convenient on your board.

(3) In the USB-Blaster and ByteBlaster II cables, this pin is connected to nCE when it is used for AS programming; otherwise it is a no connect.

(4) The nCE must be connected to GND or driven low for successful JTAG configuration.

(5) The nCEO pin is left unconnected or used as a user I/O pin when it does not feed the nCE pin of another device.

(6) Power up the V_{CC} of the EthernetBlaster, ByteBlaster II or USB-Blaster cable with supply from V_{CCIO}. The Ethernet-Blaster, ByteBlaster II, and USB-Blaster cables do not support a target supply voltage of 1.2 V. For the target supply voltage value, refer to the *ByteBlaster II Download Cable User Guide*, the *USB-Blaster Download Cable User Guide*, and the *EthernetBlaster Communications Cable User Guide*.

(7) Resistor value can vary from 1 kΩ to 10 kΩ..

To configure a single device in a JTAG chain, the programming software places all other devices in bypass mode. In bypass mode, devices pass programming data from the TDI pin to the TDO pin through a single bypass register without being affected internally. This scheme enables the programming software to program or verify the target device. Configuration data driven into the device appears on the TDO pin one clock cycle later.

The Quartus II software verifies successful JTAG configuration after completion. At the end of configuration, the software checks the state of CONF_DONE through the JTAG port. When Quartus II generates a **.jam** for a multi-device chain, it contains instructions so that all the devices in the chain are initialized at the same time. If CONF_DONE is not high, the Quartus II software indicates that configuration has failed. If CONF_DONE is high, the software indicates that configuration was successful. After the configuration bitstream is serially sent using the JTAG TDI port, the TCK port clocks an additional clock cycles to perform device initialization.

You can perform JTAG testing on Cyclone IV devices before, during, and after configuration. Cyclone IV devices support the BYPASS, IDCODE, and SAMPLE instructions during configuration without interrupting configuration. All other JTAG instructions can only be issued by first interrupting configuration and reprogramming I/O pins with the ACTIVE_DISENGAGE and CONFIG_IO instructions.

The CONFIG_IO instruction allows you to configure the I/O buffers through the JTAG port and interrupts configuration when issued after the ACTIVE_DISENGAGE instruction. This instruction allows you to perform board-level testing prior to configuring the Cyclone IV device or waiting for a configuration device to complete configuration. Prior to issuing the CONFIG_IO instruction, you must issue the ACTIVE_DISENGAGE instruction. This is because in Cyclone IV devices, the CONFIG_IO instruction does not hold nSTATUS low until reconfiguration, so you must disengage the active configuration mode controller when active configuration is interrupted. The ACTIVE_DISENGAGE instruction places the active configuration mode controllers in an idle state prior to JTAG programming. Additionally, the ACTIVE_ENGAGE instruction allows you to re-engage a disengaged active configuration mode controller.

You must follow a specific flow when executing the ACTIVE_DISENGAGE, CONFIG_IO, and ACTIVE_ENGAGE JTAG instructions in Cyclone IV devices.

The chip-wide reset (DEV_CLRn) and chip-wide output enable (DEV_OE) pins in Cyclone IV devices do not affect JTAG boundary-scan or programming operations. Toggling these pins do not affect JTAG operations (other than the usual boundary-scan operation).

When designing a board for JTAG configuration of Cyclone IV devices, consider the dedicated configuration pins. Table 8–13 describes how you must connect these pins during JTAG configuration.

Table 8–13. Dedicated Configuration Pin Connections During JTAG Configuration

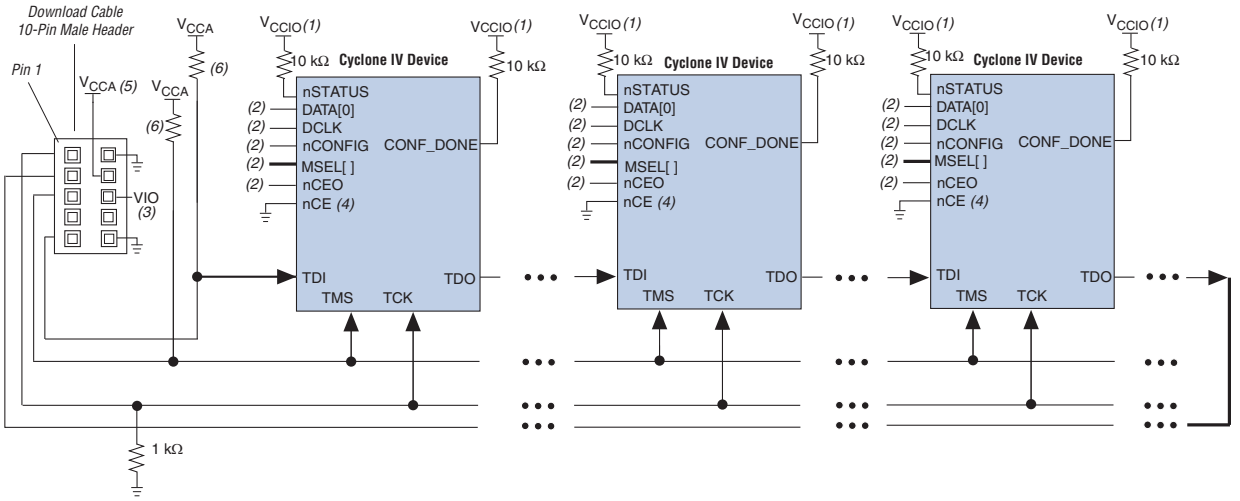
Signal	Description
nCE	On all Cyclone IV devices in the chain, nCE must be driven low by connecting it to GND, pulling it low through a resistor, or driving it by some control circuitry. For devices that are also in multi-device AS, AP, PS, or FPP configuration chains, you must connect the nCE pins to GND during JTAG configuration or JTAG configured in the same order as the configuration chain.
nCEO	On all Cyclone IV devices in the chain, nCEO is left floating or connected to the nCE of the next device.
MSEL	These pins must not be left floating. These pins support whichever non-JTAG configuration that you used in production. If you only use JTAG configuration, tie these pins to GND.
nCONFIG	Driven high by connecting to the V_{CCIO} supply of the bank in which the pin resides and pulling up through a resistor or driven high by some control circuitry.
nSTATUS	Pull to the V_{CCIO} supply of the bank in which the pin resides through a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each nSTATUS pin must be pulled up to the V_{CCIO} individually.
CONF_DONE	Pull to the V_{CCIO} supply of the bank in which the pin resides through a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each CONF_DONE pin must be pulled up to V_{CCIO} supply of the bank in which the pin resides individually. CONF_DONE going high at the end of JTAG configuration indicates successful configuration.
DCLK	Must not be left floating. Drive low or high, whichever is more convenient on your board.

When programming a JTAG device chain, one JTAG-compatible header is connected to several devices. The number of devices in the JTAG chain is limited only by the drive capability of the download cable. When four or more devices are connected in a JTAG chain, Altera recommends buffering the TCK, TDI, and TMS pins with an on-board buffer.

JTAG-chain device programming is ideal when the system contains multiple devices, or when testing your system with JTAG BST circuitry. Figure 8–25 and Figure 8–26 show multi-device JTAG configuration.

For devices using 2.5-, 3.0-, and 3.3-V V_{CCIO} supply, you must refer to Figure 8–25. All I/O inputs must maintain a maximum AC voltage of 4.1 V because JTAG pins do not have the internal PCI clamping diodes to prevent voltage overshoot when using 2.5-, 3.0-, and 3.3- V V_{CCIO} supply. You must power up the V_{CC} of the download cable with a 2.5-V V_{CCA} supply. For device using V_{CCIO} of 1.2, 1.5 V, and 1.8 V, refer to Figure 8–26. You can power up the V_{CC} of the download cable with the supply from V_{CCIO}.

Figure 8–25. JTAG Configuration of Multiple Devices Using a Download Cable (2.5, 3.0, and 3.3-V V_{CCIO} Powering the JTAG Pins)



Notes to Figure 8–25:

(1) Connect these pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) Connect the nCONFIG and MSEL pins to support a non-JTAG configuration scheme. If you only use a JTAG configuration, connect the nCONFIG pin to logic-high and the MSEL pins to GND. In addition, pull DCLK and DATA[0] to either high or low, whichever is convenient on your board.

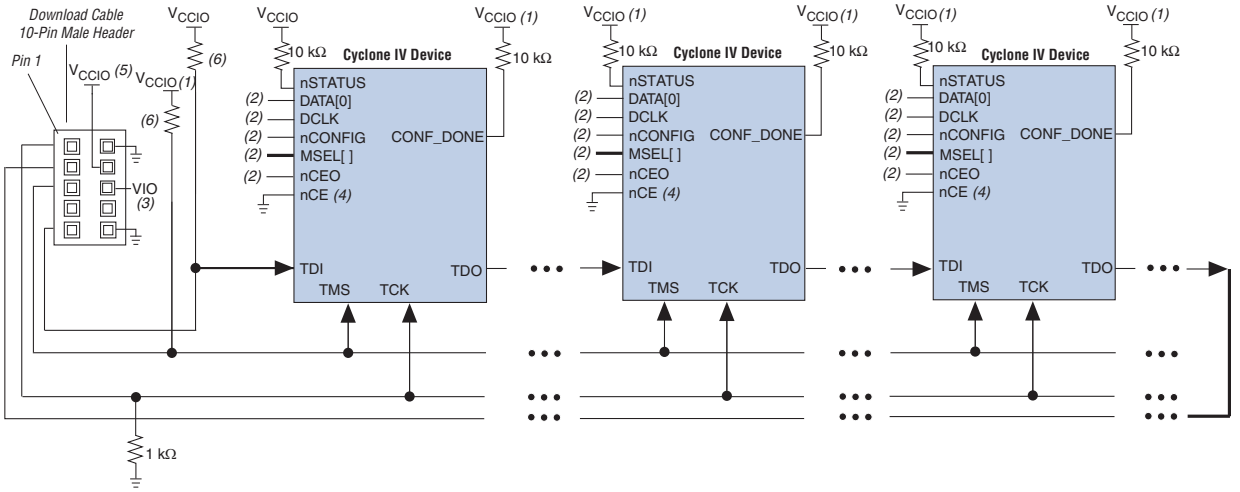
(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} must match the V_{CCA} of the device. For this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*. In the ByteBlasterMV cable, this pin is a no connect. In the USB-Blaster and ByteBlaster II cables, this pin is connected to nCE when it is used for AS programming, otherwise it is a no connect.

(4) You must connect the nCE pin to GND or driven low for successful JTAG configuration.

(5) Power up the V_{CC} of the ByteBlaster II, USB-Blaster, or ByteBlasterMV cable with a 2.5-V supply from V_{CCA}. Third-party programmers must switch to 2.5 V. Pin 4 of the header is a V_{CC} power supply for the MasterBlaster cable. The MasterBlaster cable can receive power from either 5.0- or 3.3-V circuit boards, DC power supply, or 5.0 V from the USB cable. For this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*.

(6) Resistor value can vary from 1 kΩ to 10 kΩ..

Figure 8–26. JTAG Configuration of Multiple Devices Using a Download Cable (1.2, 1.5, and 1.8-V V_{CCIO} Powering the JTAG Pins)



Notes to Figure 8–26:

(1) Connect these pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) Connect the nCONFIG and MSEL pins to support a non-JTAG configuration scheme. If you only use a JTAG configuration, connect the nCONFIG pin to logic-high and the MSEL pins to GND. In addition, pull DCLK and DATA[0] to either high or low, whichever is convenient on your board.

(3) In the USB-Blaster and ByteBlaster II cable, this pin is connected to nCE when it is used for AS programming, otherwise it is a no connect.

(4) You must connect the nCE pin to GND or driven low for successful JTAG configuration.

(5) Power up the V_{CC} of the ByteBlaster II or USB-Blaster cable with supply from V_{CCIO}. The ByteBlaster II and USB-Blaster cables do not support a target supply voltage of 1.2 V. For the target supply voltage value, refer to the *ByteBlaster II Download Cable User Guide* and the *USB-Blaster Download Cable User Guide*.

(6) Resistor value can vary from 1 kΩ to 10 kΩ..

If a non-Cyclone IV device is cascaded in the JTAG-chain, TDO of the non-Cyclone IV device driving into TDI of the Cyclone IV device must fit the maximum overshoot outlined in Equation 8–1 on page 8–5.

The CONF_DONE and nSTATUS signals are shared in multi-device AS, AP, PS, and FPP configuration chains to ensure that the devices enter user mode at the same time after configuration is complete. When the CONF_DONE and nSTATUS signals are shared among all the devices, you must configure every device when JTAG configuration is performed.

If you only use JTAG configuration, Altera recommends that you connect the circuitry as shown in Figure 8–25 or Figure 8–26, in which each of the CONF_DONE and nSTATUS signals are isolated so that each device can enter user mode individually.

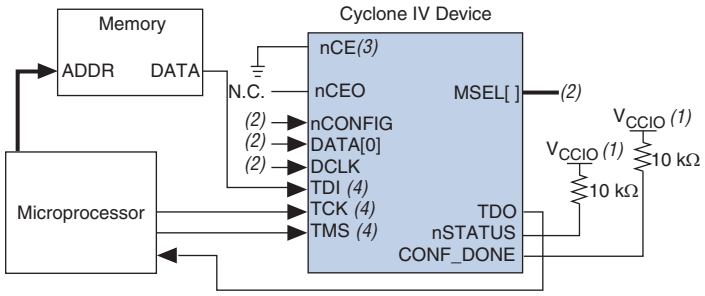
After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the nCE pin of the second device, which prompts the second device to begin configuration. Therefore, if these devices are also in a JTAG chain, ensure that the nCE pins are connected to GND during JTAG configuration or that the devices are JTAG configured in the same order as the configuration chain. As long as the devices are JTAG configured in the same order as the multi-device configuration chain, the nCEO of the previous device drives the nCE pin of the next device low when it has successfully been JTAG configured. You can place other Altera devices that have JTAG support in the same JTAG chain for device programming and configuration.

JTAG configuration allows an unlimited number of Cyclone IV devices to be cascaded in a JTAG chain.

For more information about configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook*.

Figure 8–27 shows JTAG configuration with a Cyclone IV device and a microprocessor.

Figure 8–27. JTAG Configuration of a Single Device Using a Microprocessor



Notes to Figure 8–27:

(1) You must connect the pull-up resistor to a supply that provides an acceptable input signal for all devices in the chain.

(2) Connect the `nCONFIG` and `MSEL` pins to support a non-JTAG configuration scheme. If you only use a JTAG configuration, connect the `nCONFIG` pin to logic-high and the `MSEL` pins to GND. In addition, pull `DCLK` and `DATA[0]` to either high or low, whichever is convenient on your board.

(3) You must connect the `nCE` pin to GND or driven low for successful JTAG configuration.

(4) All I/O inputs must maintain a maximum AC voltage of 4.1 V. Signals driving into `TDI`, `TMS`, and `TCK` must fit the maximum overshoot outlined in Equation 8–1 on page 8–5.

Configuring Cyclone IV Devices with Jam STAPL

Jam™ STAPL, JEDEC standard JESD-71, is a standard file format for in-system programmability (ISP) purposes. Jam STAPL supports programming or configuration of programmable devices and testing of electronic systems, using the IEEE 1149.1 JTAG interface. Jam STAPL is a freely licensed open standard. The Jam Player provides an interface for manipulating the IEEE Std. 1149.1 JTAG TAP state machine.

For more information about JTAG and Jam STAPL in embedded environments, refer to *AN 425: Using Command-Line Jam STAPL Solution for Device Programming*. To download the Jam Player, visit the Altera website (www.altera.com).

Configuring Cyclone IV Devices with the JRunner Software Driver

The JRunner software driver allows you to configure Cyclone IV devices through the ByteBlaster II or ByteBlasterMV cables in JTAG mode. The supported programming input file is in **.rbf** format. The JRunner software driver also requires a Chain Description File (**.cdf**) generated by the Quartus II software. The JRunner software driver is targeted for embedded JTAG configuration. The source code is developed for the Windows NT operating system (OS). You can customize the code to make it run on your embedded platform.

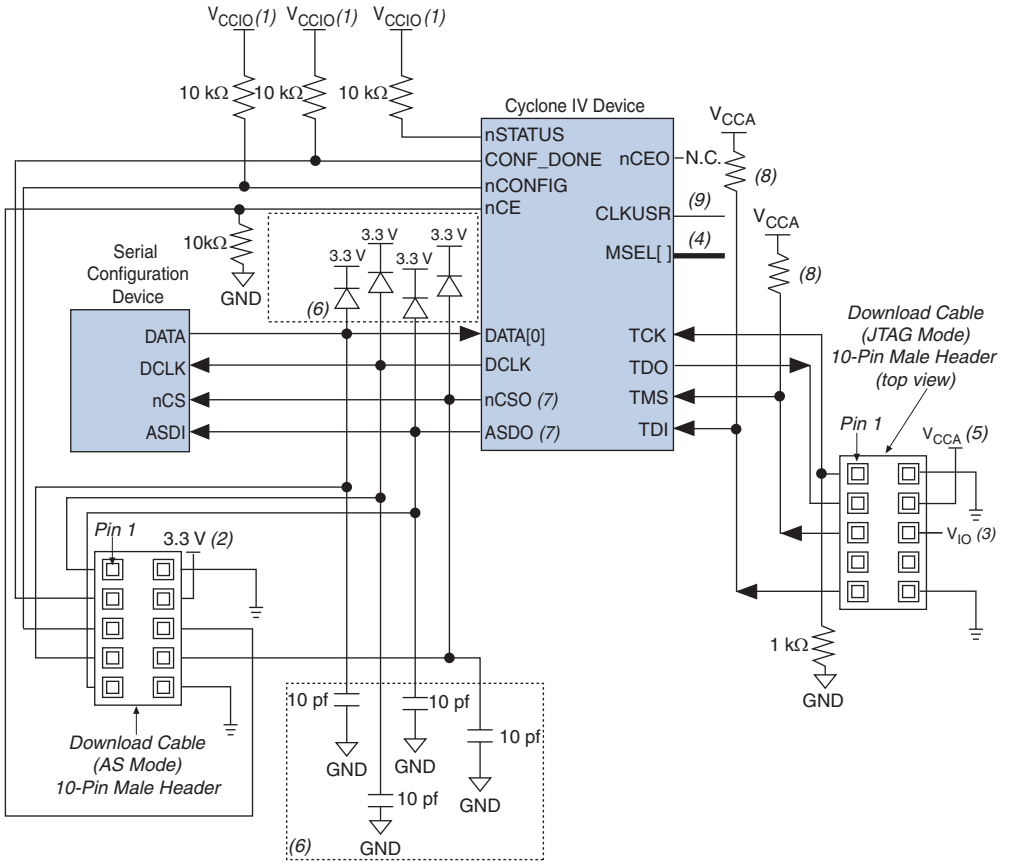
The **.rbf** used by the JRunner software driver cannot be a compressed **.rbf** because the JRunner software driver uses JTAG-based configuration. During JTAG-based configuration, the real-time decompression feature is not available.

For more information about the JRunner software driver, refer to *AN 414: JRunner Software Driver: An Embedded Solution for PLD JTAG Configuration* and the source files on the Altera website at (www.altera.com).

Combining JTAG and AS Configuration Schemes

You can combine the AS configuration scheme with the JTAG-based configuration (Figure 8–28). This setup uses two 10-pin download cable headers on the board. One download cable is used in JTAG mode to configure the Cyclone IV device directly through the JTAG interface. The other download cable is used in AS mode to program the serial configuration device in-system through the AS programming interface. If you try configuring the device using both schemes simultaneously, JTAG configuration takes precedence and AS configuration terminates.

Figure 8–28. Combining JTAG and AS Configuration Schemes



Notes to Figure 8–28:

(1) Connect these pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) Power up the V_{CC} of the EthernetBlaster, ByteBlaster II, or USB-Blaster cable with the 3.3-V supply.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver.The V_{IO} must match the V_{CCA} of the device. For this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*. When using the ByteBlasterMV download cable, this pin is a no connect. When using the USB-Blaster and ByteBlaster II cables, this pin is connected to `nCE` when it is used for AS programming, otherwise it is a no connect.

(4) The `MSEL` pin settings vary for different configuration voltage standards and POR time. To connect `MSEL` for AS configuration schemes, refer to Table 8–3 on page 8–8, Table 8–4 on page 8–8, and Table 8–5 on page 8–9. Connect the `MSEL` pins directly to V_{CCA} or GND.

(5) Power up the V_{CC} of the EthernetBlaster, ByteBlaster II, USB-Blaster, or ByteBlasterMV cable with a 2.5-V V_{CCA} supply. Third-party programmers must switch to 2.5 V. Pin 4 of the header is a V_{CC} power supply for the MasterBlaster cable. The MasterBlaster cable can receive power from either 5.0- or 3.3-V circuit boards, DC power supply, or 5.0 V from the USB cable. For this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*.

(6) You must place the diodes and capacitors as close as possible to the Cyclone IV device. Altera recommends using the Schottky diode, which has a relatively lower forward diode voltage (VF) than the switching and Zener diodes, for effective voltage clamping.

(7) These pins are dual-purpose I/O pins. The `nCSO` pin functions as `FLASH_nCE` pin in AP mode. The `ASDO` pin functions as `DATA[1]` pin in AP and FPP modes.

(8) Resistor value can vary from 1 kΩ to 10 kΩ..

(9) Only Cyclone IV GX devices have an option to select `CLKUSR` (40 MHz maximum) as the external clock source for `DCLK`.

Programming Serial Configuration Devices In-System with the JTAG Interface

Cyclone IV devices in a single- or multiple-device chain support in-system programming of a serial configuration device with the JTAG interface through the SFL design. The intelligent host or download cable of the board can use the four JTAG pins on the Cyclone IV device to program the serial configuration device in system, even if the host or download cable cannot access the configuration pins (DCLK, DATA, ASDI, and nCS pins).

The SFL design is a JTAG-based in-system programming solution for Altera serial configuration devices. The SFL is a bridge design for the Cyclone IV device that uses their JTAG interface to access the EPCS JTAG Indirect Configuration Device Programming (**.jic**) file and then uses the AS interface to program the EPCS device. Both the JTAG interface and AS interface are bridged together inside the SFL design.

In a multiple device chain, you must only configure the master device that controls the serial configuration device. Slave devices in the multiple device chain that are configured by the serial configuration device do not have to be configured when using this feature. To successfully use this feature, set the MSEL pins of the master device to select the AS configuration scheme (Table 8–3 on page 8–8, Table 8–4 on page 8–8, and Table 8–5 on page 8–9). The serial configuration device in-system programming through the Cyclone IV device JTAG interface has three stages, which are described in the following sections:

- "Loading the SFL Design"
- "ISP of the Configuration Device" on page 8–55
- "Reconfiguration" on page 8–56

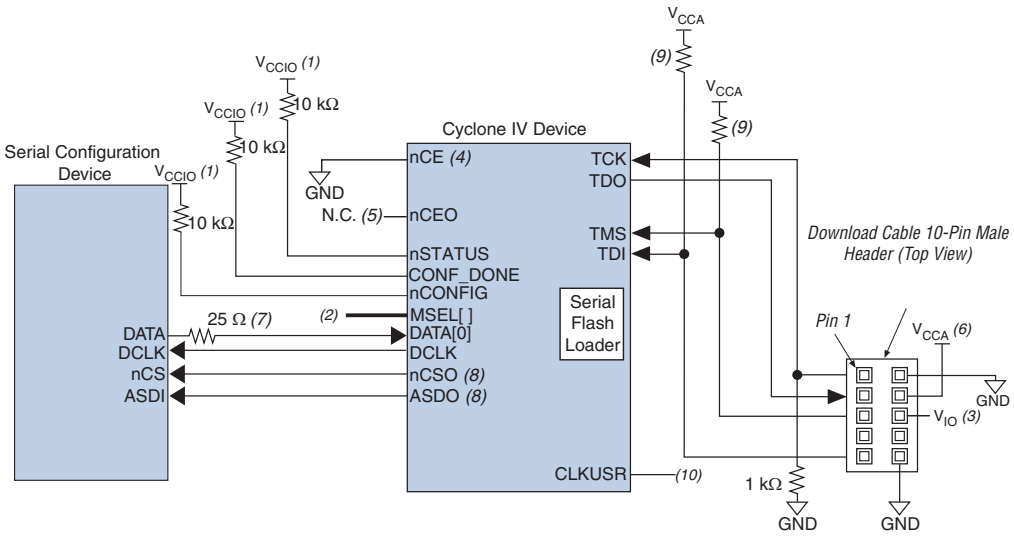
Loading the SFL Design

The SFL design is a design inside the Cyclone IV device that bridges the JTAG interface and AS interface with glue logic.

The intelligent host uses the JTAG interface to configure the master device with a SFL design. The SFL design allows the master device to control the access of four serial configuration device pins, also known as the Active Serial Memory Interface (ASMI) pins, through the JTAG interface. The ASMI pins are serial clock input (DCLK), serial data output (DATA), AS data input (ASDI), and active-low chip select (nCS) pins.

If you configure a master device with an SFL design, the master device enters user mode even though the slave devices in the multiple device chain are not being configured. The master device enters user mode with a SFL design even though the CONF_DONE signal is externally held low by the other slave devices in chain. Figure 8–29 shows the JTAG configuration of a single Cyclone IV device with a SFL design.

Figure 8–29. Programming Serial Configuration Devices In-System Using the JTAG Interface



Notes to Figure 8–29:

(1) Connect the pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) The MSEL pin settings vary for different configuration voltage standards and POR time. To connect MSEL for AS configuration schemes, refer to Table 8–3 on page 8–8, Table 8–4 on page 8–8, and Table 8–5 on page 8–9. Connect the MSEL pins directly to V_{CCA} or GND.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. The V_{IO} must match the V_{CCA} of the device. For this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*. When using the ByteBlasterMV download cable, this pin is a no connect. When using USB-Blaster, ByteBlaster II, and EthernetBlaster cables, this pin is connected to nCE when it is used for AS programming, otherwise it is a no connect.

(4) You must connect the nCE pin to GND or driven low for successful JTAG configuration.

(5) The nCEO pin is left unconnected or used as a user I/O pin when it does not feed the nCE pin of another device.

(6) Power up the V_{CC} of the EthernetBlaster, ByteBlaster II, USB-Blaster, or ByteBlasterMV cable with a 2.5- V V_{CCA} supply. Third-party programmers must switch to 2.5 V. Pin 4 of the header is a V_{CC} power supply for the MasterBlaster cable. The MasterBlaster cable can receive power from either 5.0- or 3.3-V circuit boards, DC power supply, or 5.0 V from the USB cable. For this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*.

(7) Connect the series resistor at the near end of the serial configuration device.

(8) These pins are dual-purpose I/O pins. The nCSO pin functions as FLASH_nCE pin in AP mode. The ASDO pin functions as DATA[1] pin in AP and FPP modes.

(9) Resistor value can vary from 1 kΩ to 10 kΩ..

(10) Only Cyclone IV GX devices have an option to select CLKUSR (40 MHz maximum) as the external clock source for DCLK.

ISP of the Configuration Device

In the second stage, the SFL design in the master device allows you to write the configuration data for the device chain into the serial configuration device with the Cyclone IV device JTAG interface. The JTAG interface sends the programming data for the serial configuration device to the Cyclone IV device first. The Cyclone IV device then uses the ASMI pins to send the data to the serial configuration device.

Reconfiguration

After the configuration data is successfully written into the serial configuration device, the Cyclone IV device does not automatically start reconfiguration. The intelligent host issues the PULSE_NCONFIG JTAG instruction to initialize the reconfiguration process. During reconfiguration, the master device is reset and the SFL design no longer exists in the Cyclone IV device and the serial configuration device configures all the devices in the chain with the user design.

For more information about the SFL, refer to *AN 370: Using the Serial FlashLoader with Quartus II Software.*

JTAG Instructions

For more information about the JTAG binary instruction code, refer to the *JTAG Boundary-Scan Testing for Cyclone IV Devices* chapter.

I/O Reconfiguration

Use the CONFIG_IO instruction to reconfigure the I/O configuration shift register (IOCSR) chain. This instruction allows you to perform board-level testing prior to configuring the Cyclone IV device or waiting for a configuration device to complete configuration. After the configuration is interrupted and JTAG testing is complete, you must reconfigure the part through the PULSE_NCONFIG JTAG instruction or by pulsing the nCONFIG pin low.

You can issue the CONFIG_IO instruction any time during user mode.

You must meet the following timing restrictions when using the CONFIG_IO instruction:

- The CONFIG_IO instruction cannot be issued when the nCONFIG pin is low
- You must observe a 230 µs minimum wait time after any of the following conditions:
  - nCONFIG pin goes high
  - Issuing the PULSE_NCONFIG instruction
  - Issuing the ACTIVE_ENGAGE instruction, before issuing the CONFIG_IO instruction
- You must wait 230 µs after power up, with the nCONFIG pin high before issuing the CONFIG_IO instruction (or wait for the nSTATUS pin to go high)

Use the ACTIVE_DISENGAGE instruction with the CONFIG_IO instruction to interrupt configuration. Table 8–14 lists the sequence of instructions to use for various CONFIG_IO usage scenarios.

Table 8–14. JTAG CONFIG_IO (without JTAG_PROGRAM) Instruction Flows *(Note 1)*

JTAG Instruction	Configuration Scheme and Current State of the Cyclone IV Device											
	Prior to User Mode (Interrupting Configuration)				User Mode				Power Up			
	PS	FPP	AS	AP	PS	FPP	AS	AP	PS	FPP	AS	AP
ACTIVE_DISENGAGE	O	O	O	O	O	O	O	O	—	—	—	—
CONFIG_IO	R	R	R	R	R	R	R	R	NA	NA	NA	NA
JTAG Boundary Scan Instructions (no JTAG_PROGRAM)	O	O	O	O	O	O	O	O	—	—	—	—
ACTIVE_ENGAGE	A	A	R *(2)*	R *(2)*	A	A	R *(2)*	R *(2)*	—	—	—	—
PULSE_NCONFIG			A *(3)*	A *(3)*			O	O	—	—	—	—
Pulse nCONFIG pin			A *(3)*	A *(3)*			O	O	—	—	—	—
JTAG TAP Reset	R	R	R	R	R	R	R	R	—	—	—	—

Notes to Table 8–14:

(1) You must execute "R" indicates that the instruction before the next instruction, "O" indicates the optional instruction, "A" indicates that the instruction must be executed, and "NA" indicates that the instruction is not allowed in this mode.

(2) Required if you use ACTIVE_DISENGAGE.

(3) Neither of the instruction is required if you use ACTIVE_ENGAGE.

The CONFIG_IO instruction does not hold nSTATUS low until reconfiguration. You must disengage the AS or AP configuration controller by issuing the ACTIVE_DISENGAGE and ACTIVE_ENGAGE instructions when active configuration is interrupted. You must issue the ACTIVE_DISENGAGE instruction alone or prior to the CONFIG_IO instruction if the JTAG_PROGRAM instruction is to be issued later (Table 8–15). This puts the active configuration controllers into the idle state. The active configuration controller is re-engaged after user mode is reached through JTAG programming (Table 8–15).

While executing the CONFIG_IO instruction, all user I/Os are tri-stated.

If reconfiguration after interruption is performed using configuration modes (rather than using JTAG_PROGRAM), it is not necessary to issue the ACTIVE_DISENGAGE instruction prior to CONFIG_IO. You can start reconfiguration by either pulling nCONFIG low for at least 500 ns or issuing the PULSE_NCONFIG instruction. If the ACTIVE_DISENGAGE instruction was issued and the JTAG_PROGRAM instruction fails to enter user mode, you must issue the ACTIVE_ENGAGE instruction to reactivate the active configuration controller. Issuing the ACTIVE_ENGAGE instruction also triggers reconfiguration in configuration modes; therefore, it is not necessary to pull nCONFIG low or issue the PULSE_NCONFIG instruction.

ACTIVE_DISENGAGE

The ACTIVE_DISENGAGE instruction places the active configuration controller (AS and AP) into an idle state prior to JTAG programming. The two purposes of placing the active controller in an idle state are:

- To ensure that it is not trying to configure the device during JTAG programming

- To allow the controllers to properly recognize a successful JTAG programming that results in the device reaching user mode

The ACTIVE_DISENGAGE instruction is required before JTAG programming regardless of the current state of the Cyclone IV device if the MSEL pins are set to an AS or AP configuration scheme. If the ACTIVE_DISENGAGE instruction is issued during a passive configuration scheme (PS or FPP), it has no effect on the Cyclone IV device. Similarly, the CONFIG_IO instruction is issued after an ACTIVE_DISENGAGE instruction, but is no longer required to properly halt configuration. Table 8–15 lists the required, recommended, and optional instructions for each configuration mode. The ordering of the required instructions is a hard requirement and must be met to ensure functionality.

Table 8–15. JTAG Programming Instruction Flows *(Note 1)*

JTAG Instruction	Configuration Scheme and Current State of the Cyclone IV Device											
	Prior to User Mode (Interrupting Configuration)				User Mode				Power Up			
	PS	FPP	AS	AP	PS	FPP	AS	AP	PS	FPP	AS	AP
ACTIVE_DISENGAGE	0	0	R	R	0	0	0	R	0	0	R	R
CONFIG_IO	Rc	Rc	0	0	0	0	0	0	NA	NA	NA	NA
Other JTAG instructions	0	0	0	0	0	0	0	0	0	0	0	0
JTAG_PROGRAM	R	R	R	R	R	R	R	R	R	R	R	R
CHECK_STATUS	Rc	Rc	Rc	Rc	Rc	Rc	Rc	Rc	Rc	Rc	Rc	Rc
JTAG_STARTUP	R	R	R	R	R	R	R	R	R	R	R	R
JTAG TAP Reset/other instruction	R	R	R	R	R	R	R	R	R	R	R	R

Note to Table 8–15:

(1) "R" indicates that the instruction must be executed before the next instruction, "O" indicates the optional instruction, "Rc" indicates the recommended instruction, and "NA" indicates that the instruction is not allowed in this mode.

In the AS or AP configuration scheme, the ACTIVE_DISENGAGE instruction puts the active configuration controller into idle state. If a successful JTAG programming is executed, the active controller is automatically re-engaged after user mode is reached through JTAG programming. This causes the active controller to transition to their respective user mode states.

If JTAG programming fails to get the Cyclone IV device to enter user mode and re-engage active programming, there are available methods to achieve this:

- In AS configuration scheme, you can re-engage the AS controller by moving the JTAG TAP controller to the reset state or by issuing the ACTIVE_ENGAGE instruction.
- In AP configuration scheme, the only way to re-engage the AP controller is to issue the ACTIVE_ENGAGE instruction. In this case, asserting the nCONFIG pin does not re-engage either active controller.

ACTIVE_ENGAGE

The ACTIVE_ENGAGE instruction allows you to re-engage a disengaged active controller. You can issue this instruction any time during configuration or user mode to re-engage an already disengaged active controller, as well as trigger reconfiguration of the Cyclone IV device in the active configuration scheme.

The ACTIVE_ENGAGE instruction functions as the PULSE_NCONFIG instruction when the device is in the PS or FPP configuration schemes. The nCONFIG pin is disabled when the ACTIVE_ENGAGE instruction is issued.

Altera does not recommend using the ACTIVE_ENGAGE instruction, but it is provided as a fail-safe instruction for re-engaging the active configuration controller (AS and AP).

Overriding the Internal Oscillator

This feature allows you to override the internal oscillator during the active configuration scheme. The AS and AP configuration controllers use the internal oscillator as the clock source. You can change the clock source to CLKUSR through the JTAG instruction.

The EN_ACTIVE_CLK and DIS_ACTIVE_CLK JTAG instructions toggle on or off whether or not the active clock is sourced from the CLKUSR pin or the internal configuration oscillator. To source the active clock from the CLKUSR pin, issue the EN_ACTIVE_CLK instruction. This causes the CLKUSR pin to become the active clock source. When using the EN_ACTIVE_CLK instruction, you must enable the internal oscillator for the clock change to occur. By default, the configuration oscillator is disabled after configuration and initialization is complete as well as the device has entered user mode.

However, the internal oscillator is enabled in user mode by any of the following conditions:

- A reconfiguration event (for example, driving the nCONFIG pin to go low)
- Remote update is enabled
- Error detection is enabled

You must clock the CLKUSR pin at two times the expected DCLK frequency. The CLKUSR pin allows a maximum frequency of 80 MHz (40 MHz DCLK). Normally, a test instrument uses the CLKUSR pin when it wants to drive its own clock to control the AS state machine.

To revert the clock source back to the configuration oscillator, issue the DIS_ACTIVE_CLK instruction. After you issue the DIS_ACTIVE_CLK instruction, you must continue to clock the CLKUSR pin for 10 clock cycles. Otherwise, even toggling the nCONFIG pin does not revert the clock source and reconfiguration does not occur. A POR reverts the clock source back to the configuration oscillator. Toggling the nCONFIG pin or driving the JTAG state machine to reset state does not revert the clock source.

EN_ACTIVE_CLK

The EN_ACTIVE_CLK instruction causes the CLKUSR pin signal to replace the internal oscillator as the clock source. When using the EN_ACTIVE_CLK instruction, you must enable the internal oscillator for the clock change to occur. After this instruction is issued, other JTAG instructions can be issued while the CLKUSR pin signal remains as the clock source. The clock source is only reverted back to the internal oscillator by issuing the DIS_ACTIVE_CLK instruction or a POR.

DIS_ACTIVE_CLK

The DIS_ACTIVE_CLK instruction breaks the CLKUSR enable latch set by the EN_ACTIVE_CLK instruction and causes the clock source to revert back to the internal oscillator. After the DIS_ACTIVE_CLK instruction is issued, you must continue to clock the CLKUSR pin for 10 clock cycles.

You must clock the CLKUSR pin at two times the expected DCLK frequency. The CLKUSR pin allows a maximum frequency of 80 MHz (40 MHz DCLK).

Changing the Start Boot Address of the AP Flash

In the AP configuration scheme (for Cyclone IV E devices only), you can change the default configuration boot address of the parallel flash memory to any desired address using the APFC_BOOT_ADDR JTAG instruction.

APFC_BOOT_ADDR

The APFC_BOOT_ADDR instruction is for Cyclone IV E devices only and allows you to define a start boot address for the parallel flash memory in the AP configuration scheme.

This instruction shifts in a start boot address for the AP flash. When this instruction becomes the active instruction, the TDI and TDO pins are connected through a 22-bit active boot address shift register. The shifted-in boot address bits get loaded into the 22-bit AP boot address update register, which feeds into the AP controller. The content of the AP boot address update register can be captured and shifted-out of the active boot address shift register from TDO.

The boot address in the boot address shift register and update register are shifted to the right (in the LSB direction) by two bits versus the intended boot address. The reason for this is that the two LSB of the address are not accessible. When this boot address is fed into the AP controller, two 0s are attached in the end as LSB, thereby pushing the shifted-in boot address to the left by two bits, which become the actual AP boot address the AP controller gets.

If you have enabled the remote update feature, the APFC_BOOT_ADDR instruction sets the boot address for the factory configuration only.

The APFC_BOOT_ADDR instruction is retained after reconfiguration while the system board is still powered on. However, you must reprogram the instruction whenever you restart the system board.

Device Configuration Pins

Table 8–16 through Table 8–19 describe the connections and functionality of all the configuration related pins on Cyclone IV devices. Table 8–16 and Table 8–17 list the device pin configuration for the Cyclone IV GX and Cyclone IV E, respectively.

Table 8–16. Configuration Pin Summary for Cyclone IV GX Devices

Bank	Description	Input/Output	Dedicated	Powered By	Configuration Mode
8	Data[4:2]	Input	—	V_{CCIO}	FPP
3	Data[7:5]	Input	—	V_{CCIO}	FPP
9	nCSO *(2)*	Output	—	V_{CCIO}	AS
3	CRC_ERROR	Output	—	V_{CCIO}/Pull-up *(1)*	Optional, all modes
9	DATA[0] *(2)*	Input	Yes	V_{CCIO}	PS, FPP, AS
9	DATA[1]/ASDO *(2)*	Input	—	V_{CCIO}	FPP
		Output		V_{CCIO}	AS
3	INIT_DONE	Output	—	Pull-up	Optional, all modes
3	nSTATUS	Bidirectional	Yes	Pull-up	All modes
9	nCE	Input	Yes	V_{CCIO}	All modes
9	DCLK *(2)*	Input	Yes	V_{CCIO}	PS, FPP
		Output		V_{CCIO}	AS
3	CONF_DONE	Bidirectional	Yes	Pull-up	All modes
9	TDI	Input	Yes	V_{CCIO}	JTAG
9	TMS	Input	Yes	V_{CCIO}	JTAG
9	TCK	Input	Yes	V_{CCIO}	JTAG
9	nCONFIG	Input	Yes	V_{CCIO}	All modes
8	CLKUSR	Input	—	V_{CCIO}	Optional
3	nCEO	Output	—	V_{CCIO}	Optional, all modes
3	MSEL	Input	Yes	V_{CCINT}	All modes
9	TDO	Output	Yes	V_{CCIO}	JTAG
6	DEV_OE	Input	—	V_{CCIO}	Optional
6	DEV_CLRn	Input	—	V_{CCIO}	Optional

Notes to Table 8–16:

(1) The CRC_ERROR pin is a dedicated output by default. Optionally, you can enable the CRC_ERROR pin as an open-drain output in the **CRC Error Detection** tab of the **Device and Pin Options** dialog box.

(2) To tri-state AS configuration pins in the AS configuration scheme, turn on the **Enable input tri-state on active configuration pins in user mode** option from the **Device and Pin Options** dialog box. This tri-states DCLK, nCSO, Data[0], and Data[1]/ASDO pins. Dual-purpose pins settings for these pins are ignored. To set these pins to different settings, turn off the **Enable input tri-state on active configuration pins in user mode** option and set the desired setting from the Dual-purpose Pins Setting menu.

Table 8–17. Configuration Pin Summary for Cyclone IV E Devices (Part 1 of 2)

Bank	Description	Input/Output	Dedicated	Powered By	Configuration Mode
1	nCSO *(1)*, FLASH_nCE *(2)*	Output	—	V_{CCIO}	AS, AP
6	CRC_ERROR *(3)*	Output	—	V_{CCIO}/Pull-up *(4)*	Optional, all modes
1	DATA[0] *(1)*, *(2)*	Input	Yes	V_{CCIO}	PS, FPP, AS
		Bidirectional		V_{CCIO}	AP
1	DATA[1] *(2)*/ASDO *(1)*	Input	—	V_{CCIO}	FPP
		Output		V_{CCIO}	AS
		Bidirectional		V_{CCIO}	AP
8	DATA[7..2] *(2)*	Input	—	V_{CCIO}	FPP
		Bidirectional		V_{CCIO}	AP
8	DATA[15..8] *(2)*	Bidirectional	—	V_{CCIO}	AP
6	INIT_DONE	Output	—	Pull-up	Optional, all modes
1	nSTATUS	Bidirectional	Yes	Pull-up	All modes
1	nCE	Input	Yes	V_{CCIO}	All modes
1	DCLK *(1)*, *(2)*	Input	Yes	V_{CCIO}	PS, FPP
		Output		V_{CCIO}	AS, AP
6	CONF_DONE	Bidirectional	Yes	Pull-up	All modes
1	TDI	Input	Yes	V_{CCIO}	JTAG
1	TMS	Input	Yes	V_{CCIO}	JTAG
1	TCK	Input	Yes	V_{CCIO}	JTAG
1	nCONFIG	Input	Yes	V_{CCIO}	All modes
6	CLKUSR	Input	—	V_{CCIO}	Optional
6	nCEO	Output	—	V_{CCIO}	Optional, all modes
6	MSEL[]	Input	Yes	V_{CCINT}	All modes
1	TDO	Output	Yes	V_{CCIO}	JTAG
7	PADD[14..0]	Output	—	V_{CCIO}	AP
8	PADD[19..15]	Output	—	V_{CCIO}	AP
6	PADD[23..20]	Output	—	V_{CCIO}	AP
1	nRESET	Output	—	V_{CCIO}	AP
6	nAVD	Output	—	V_{CCIO}	AP
6	nOE	Output	—	V_{CCIO}	AP
6	nWE	Output	—	V_{CCIO}	AP
5	DEV_OE	Input	—	V_{CCIO}	Optional, AP

Table 8–17. Configuration Pin Summary for Cyclone IV E Devices (Part 2 of 2)

Bank	Description	Input/Output	Dedicated	Powered By	Configuration Mode
5	DEV_CLRn	Input	—	V_{CCIO}	Optional, AP

Notes to Table 8–17:

(1) To tri-state AS configuration pins in the AS configuration scheme, turn-on the **Enable input tri-state on active configuration pins in user mode** option from the **Device and Pin Options** dialog box. This tri-states DCLK, nCSO, Data[0], and Data[1]/ASDO pins. Dual-purpose pins settings for these pins are ignored. To set these pins to different settings, turn off the **Enable input tri-state on active configuration pins in user mode** option and set the desired setting from the Dual-purpose Pins Setting menu.

(2) To tri-state AP configuration pins in the AP configuration scheme, turn-on the **Enable input tri-state on active configuration pins in user mode** option from the **Device and Pin Options** dialog box. This tri-states DCLK, Data[0..15], FLASH_nCE, and other AP pins. Dual-purpose pins settings for these pins are ignored. To set these pins to different settings, turn off the **Enable input tri-state on active configuration pins in user mode** option and set the desired setting from the Dual-purpose Pins Setting menu.

(3) The CRC_ERROR pin is not available in Cyclone IV E devices with 1.0-V core voltage.

(4) The CRC_ERROR pin is a dedicated output by default. Optionally, you can enable the CRC_ERROR pin as an open-drain output in the **CRC Error Detection** tab of the **Device and Pin Options** dialog box.

Table 8–18 describes the dedicated configuration pins. You must properly connect these pins on your board for successful configuration. You may not need some of these pins for your configuration schemes.

Table 8–18. Dedicated Configuration Pins on the Cyclone IV Device (Part 1 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
MSEL	N/A	All	Input	Configuration input that sets the Cyclone IV device configuration scheme. You must hardwire these pins to V_{CCA} or GND. The MSEL pins have internal 9-kΩ pull-down resistors that are always active.
nCONFIG	N/A	All	Input	Configuration control input. Pulling this pin low with external circuitry during user mode causes the Cyclone IV device to lose its configuration data, enter a reset state, and tri-state all I/O pins. Returning this pin to a logic-high level starts a reconfiguration.
nSTATUS	N/A	All	Bidirectional open-drain	The Cyclone IV device drives nSTATUS low immediately after power-up and releases it after the POR time. ■ Status output—if an error occurs during configuration, nSTATUS is pulled low by the target device. ■ Status input—if an external source (for example, another Cyclone IV device) drives the nSTATUS pin low during configuration or initialization, the target device enters an error state. Driving nSTATUS low after configuration and initialization does not affect the configured device. If you use a configuration device, driving nSTATUS low causes the configuration device to attempt to configure the device, but because the device ignores transitions on nSTATUS in user mode, the device does not reconfigure. To start a reconfiguration, you must pull nCONFIG low.

Table 8–18. Dedicated Configuration Pins on the Cyclone IV Device (Part 2 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
CONF_DONE	N/A	All	Bidirectional open-drain	■ Status output—the target Cyclone IV device drives the CONF_DONE pin low before and during configuration. After all the configuration data is received without error and the initialization cycle starts, the target device releases CONF_DONE. ■ Status input—after all the data is received and CONF_DONE goes high, the target device initializes and enters user mode. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize. Driving CONF_DONE low after configuration and initialization does not affect the configured device. Do not connect bus holds or ADC to CONF_DONE pin.
nCE	N/A	All	Input	Active-low chip enable. The nCE pin activates the Cyclone IV device with a low signal to allow configuration. You must hold nCE pin low during configuration, initialization, and user-mode. In a single-device configuration, you must tie the nCE pin low. In a multi-device configuration, nCE of the first device is tied low while its nCEO pin is connected to nCE of the next device in the chain. You must hold the nCE pin low for successful JTAG programming of the device.
nCEO	N/A if option is on. I/O if option is off.	All	Output open-drain	Output that drives low when configuration is complete. In a single-device configuration, you can leave this pin floating or use it as a user I/O pin after configuration. In a multi-device configuration, this pin feeds the nCE pin of the next device. The nCEO of the last device in the chain is left floating or used as a user I/O pin after configuration. If you use the nCEO pin to feed the nCE pin of the next device, use an external 10-kΩ pull-up resistor to pull the nCEO pin high to the V_{CCIO} voltage of its I/O bank to help the internal weak pull-up resistor. If you use the nCEO pin as a user I/O pin after configuration, set the state of the pin on the **Dual-Purpose Pin** settings.
nCSO, FLASH_nCE	I/O	AS, AP *(1)*	Output	Output control signal from the Cyclone IV device to the serial configuration device in AS mode that enables the configuration device. This pin functions as nCSO in AS mode and FLASH_nCE in AP mode. Output control signal from the Cyclone IV device to the parallel flash in AP mode that enables the flash. Connects to the CE# pin on the Numonyx P30 or P33 flash. *(1)* This pin has an internal pull-up resistor that is always active.

Table 8–18. Dedicated Configuration Pins on the Cyclone IV Device (Part 3 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
DCLK	N/A	PS, FPP, AS, AP *(1)*	Input (PS, FPP). Output (AS, AP) *(1)*	In PS and FPP configuration, DCLK is the clock input used to clock data from an external source into the target Cyclone IV device. Data is latched into the device on the rising edge of DCLK. In AS mode, DCLK is an output from the Cyclone IV device that provides timing for the configuration interface. It has an internal pull-up resistor (typically 25 kΩ) that is always active. In AP mode, DCLK is an output from the Cyclone IV E device that provides timing for the configuration interface. *(1)* In AS or AP configuration schemes, this pin is driven into an inactive state after configuration completes. Alternatively, in active schemes, you can use this pin as a user I/O during user mode. In PS or FPP schemes that use a control host, you must drive DCLK either high or low, whichever is more convenient. In passive schemes, you cannot use DCLK as a user I/O in user mode. Toggling this pin after configuration does not affect the configured device.
DATA[0]	I/O	PS, FPP, AS, AP *(1)*	Input (PS, FPP, AS). Bidirectional (AP) *(1)*	Data input. In serial configuration modes, bit-wide configuration data is presented to the target Cyclone IV device on the DATA[0] pin. In AS mode, DATA[0] has an internal pull-up resistor that is always active. After AS configuration, DATA[0] is a dedicated input pin with optional user control. After PS or FPP configuration, DATA[0] is available as a user I/O pin. The state of this pin depends on the **Dual-Purpose Pin** settings. After AP configuration, DATA[0] is a dedicated bidirectional pin with optional user control. *(1)*
DATA[1]/ ASDO	I/O	FPP, AS, AP *(1)*	Input (FPP). Output (AS). Bidirectional (AP) *(1)*	The DATA[1] pin functions as the ASDO pin in AS mode. Data input in non-AS mode. Control signal from the Cyclone IV device to the serial configuration device in AS mode used to read out configuration data. In AS mode, DATA[1] has an internal pull-up resistor that is always active. After AS configuration, DATA[1] is a dedicated output pin with optional user control. In a PS configuration scheme, DATA[1] functions as a user I/O pin during configuration, which means it is tri-stated. After FPP configuration, DATA[1] is available as a user I/O pin and the state of this pin depends on the **Dual-Purpose Pin** settings. In an AP configuration scheme, for Cyclone IV E devices only, the byte-wide or word-wide configuration data is presented to the target Cyclone IV E device on DATA[7..0] or DATA[15..0], respectively. After AP configuration, DATA[1] is a dedicated bidirectional pin with optional user control. *(1)*

Table 8–18. Dedicated Configuration Pins on the Cyclone IV Device (Part 4 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
DATA[7..2]	I/O	FPP, AP *(1)*	Inputs (FPP). Bidirectional (AP) *(1)*	In an AS or PS configuration scheme, DATA[7..2] function as user I/O pins during configuration, which means they are tri-stated. After FPP configuration, DATA[7..2] are available as user I/O pins and the state of these pin depends on the **Dual-Purpose Pin** settings. In an AP configuration scheme, for Cyclone IV E devices only, the byte-wide or word-wide configuration data is presented to the target Cyclone IV E device on DATA[7..0] or DATA[15..0], respectively. After AP configuration, DATA[7..2] are dedicated bidirectional pins with optional user control. *(1)*
DATA[15..8]	I/O	AP *(1)*	Bidirectional	Data inputs. Word-wide configuration data is presented to the target Cyclone IV E device on DATA[15..0]. In a PS, FPP, or AS configuration scheme, DATA[15:8] function as user I/O pins during configuration, which means they are tri stated. After AP configuration, DATA[15:8] are dedicated bidirectional pins with optional user control.
PADD[23..0]	I/O	AP *(1)*	Output	In AP mode, it is a 24-bit address bus from the Cyclone IV E device to the parallel flash. Connects to the A[24:1] bus on the Numonyx P30 or P33 flash.
nRESET	I/O	AP *(1)*	Output	Active-low reset output. Driving the nRESET pin low resets the parallel flash. Connects to the RST# pin on the Numonyx P30 or P33 flash.
nAVD	I/O	AP *(1)*	Output	Active-low address valid output. Driving the nAVD pin low during read or write operation indicates to the parallel flash that a valid address is present on the PADD[23..0] address bus. Connects to the ADV# pin on the Numonyx P30 or P33 flash.
nOE	I/O	AP *(1)*	Output	Active-low output enable to the parallel flash. During the read operation, driving the nOE pin low enables the parallel flash outputs (DATA[15..0]). Connects to the OE# pin on the Numonyx P30 or P33 flash.
nWE	I/O	AP *(1)*	Output	Active-low write enable to the parallel flash. During the write operation, driving the nWE pin low indicates to the parallel flash that data on the DATA[15..0] bus is valid. Connects to the WE# pin on the Numonyx P30 or P33 flash.

Note to Table 8–18:

(1) The AP configuration scheme is for Cyclone IV E devices only.

© December 2010 Altera Corporation

Table 8–19 lists the optional configuration pins. If you do not enable these optional configuration pins in the Quartus II software, they are available as general-purpose user I/O pins. Therefore, during configuration, these pins function as user I/O pins and are tri-stated with weak pull-up resistors.

Table 8–19. Optional Configuration Pins

Pin Name	User Mode	Pin Type	Description
CLKUSR	N/A if option is on. I/O if option is off.	Input	Optional user-supplied clock input synchronizes the initialization of one or more devices. This pin is enabled by turning on the **Enable user-supplied start-up clock (CLKUSR)** option in the Quartus II software. In AS configuration for Cyclone IV GX devices, you can use this pin as an external clock source to generate the DCLK by changing the clock source option in the Quartus II software in the **Configuration** tab of the **Device and Pin Options** dialog box.
INIT_DONE	N/A if option is on. I/O if option is off.	Output open-drain	Status pin is used to indicate when the device has initialized and is in user-mode. When nCONFIG is low, the INIT_DONE pin is tri-stated and pulled high due to an external 10-kΩ pull-up resistor during the beginning of configuration. After the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high and the device enters user mode. Thus, the monitoring circuitry must be able to detect a low-to-high transition. This pin is enabled by turning on the **Enable INIT_DONE output** option in the Quartus II software. The functionality of this pin changes if the **Enable OCT_DONE** option is enabled in the Quartus II software. This option controls whether the INIT_DONE signal is gated by the OCT_DONE signal, which indicates the power-up on-chip termination (OCT) calibration is complete. If this option is turned off, the INIT_DONE signal is not gated by the OCT_DONE signal.
DEV_OE	N/A if option is on. I/O if option is off.	Input	Optional pin that allows you to override all tri-states on the device. When this pin is driven low, all I/O pins are tri-stated; when this pin is driven high, all I/O pins behave as programmed. This pin is enabled by turning on the **Enable device-wide output enable (DEV_OE)** option in the Quartus II software.
DEV_CLRn	N/A if option is on. I/O if option is off.	Input	Optional pin that allows you to override all clears on all device registers. When this pin is driven low, all registers are cleared; when this pin is driven high, all registers behave as programmed. You can enable this pin by turning on the **Enable device-wide reset (DEV_CLRn)** option in the Quartus II software.

Remote System Upgrade

Cyclone IV devices support remote system upgrade in AS and AP configuration schemes. You can also implement remote system upgrade with advanced Cyclone IV features such as real-time decompression of configuration data in the AS configuration scheme.

Remote system upgrade is not supported in a multi-device configuration chain for any configuration scheme.

Functional Description

The dedicated remote system upgrade circuitry in Cyclone IV devices manages remote configuration and provides error detection, recovery, and status information. A Nios® II processor or a user logic implemented in the Cyclone IV device logic array provides access to the remote configuration data source and an interface to the configuration memory.

Configuration memory refers to serial configuration devices (EPCS) or supported parallel flash memory, depending on the configuration scheme that is used.

The remote system upgrade process of the Cyclone IV device consists of the following steps:

1. A Nios II processor (or user logic) implemented in the Cyclone IV device logic array receives new configuration data from a remote location. The connection to the remote source is a communication protocol, such as the transmission control protocol/Internet protocol (TCP/IP), peripheral component interconnect (PCI), user datagram protocol (UDP), universal asynchronous receiver/transmitter (UART), or a proprietary interface.
2. The Nios II processor (or user logic) writes this new configuration data into a configuration memory.
3. The Nios II processor (or user logic) starts a reconfiguration cycle with the new or updated configuration data.
4. The dedicated remote system upgrade circuitry detects and recovers from any error that might occur during or after the reconfiguration cycle and provides error status information to the user design.

Figure 8–30 shows the steps required for performing remote configuration updates (the numbers in Figure 8–30 coincide with steps 1–3).

Figure 8–30. Functional Diagram of Cyclone IV Device Remote System Upgrade

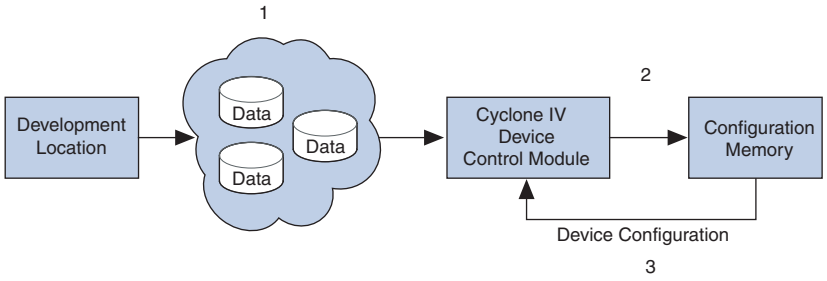
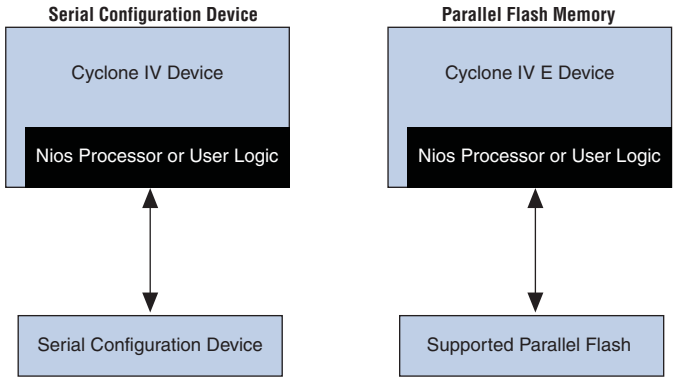


Figure 8–31 shows the block diagrams to implement remote system upgrade in Cyclone IV devices.

Figure 8–31. Remote System Upgrade Block Diagrams for AS and AP Configuration Schemes



The MSEL pin setting in the remote system upgrade mode is the same as the standard configuration mode. Standard configuration mode refers to normal Cyclone IV device configuration mode with no support for remote system upgrades (the remote system upgrade circuitry is disabled). When using remote system upgrade in Cyclone IV devices, you must enable the remote update mode option setting in the Quartus II software.

Enabling Remote Update

You can enable or disable remote update for Cyclone IV devices in the Quartus II software before design compilation (in the Compiler Settings menu). To enable remote update in the compiler settings of the project, perform the following steps:

1. On the Assignments menu, click **Device**. The **Settings** dialog box appears.
2. Click **Device and Pin Options**. The **Device and Pin Options** dialog box appears.
3. Click the **Configuration** tab.
4. From the **Configuration Mode** list, select **Remote**.
5. Click **OK**.
6. In the **Settings** dialog box, click **OK**.

Configuration Image Types

When using remote system upgrade, Cyclone IV device configuration bitstreams are classified as factory configuration images or application configuration images. An image, also referred to as a configuration, is a design loaded into the device that performs certain user-defined functions. Each device in your system requires one factory image or with addition of one or more application images. The factory image is a user-defined fall-back or safe configuration and is responsible for administering remote updates with the dedicated circuitry. Application images implement user-defined functionality in the target Cyclone IV device. You can include the default application image functionality in the factory image.

Remote System Upgrade Mode

In remote update mode, Cyclone IV devices load the factory configuration image after power up. The user-defined factory configuration determines the application configuration to be loaded and triggers a reconfiguration cycle. The factory configuration can also contain application logic.

When used with configuration memory, the remote update mode allows an application configuration to start at any flash sector boundary. Additionally, the remote update mode features a user watchdog timer that can detect functional errors in an application configuration.

Remote Update Mode

In AS configuration scheme, when a Cyclone IV device is first powered up in remote update, it loads the factory configuration located at address `boot_address[23:0] = 24b'0`. Altera recommends storing the factory configuration image for your system at boot address `24b'0`, which corresponds to the start address location `0×000000` in the serial configuration device. A factory configuration image is a bitstream for the Cyclone IV device in your system that is programmed during production and is the fall-back image when an error occurs. This image is stored in non-volatile memory and is never updated or modified using remote access.

When you use the AP configuration in Cyclone IV E devices, the Cyclone IV E device loads the default factory configuration located at the following address after device power-up in remote update mode:
`boot_address[23:0] = 24'h010000 = 24'b1 0000 0000 0000 0000`.

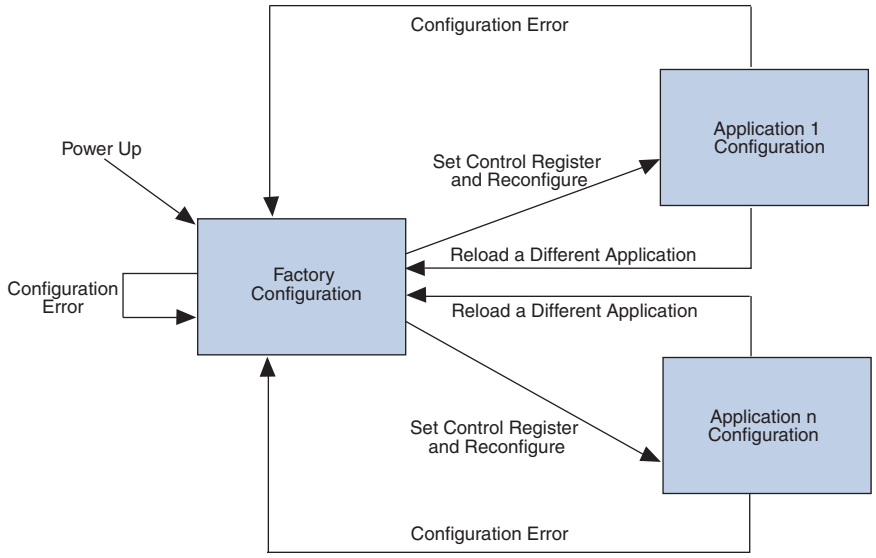
You can change the default factory configuration address to any desired address using the `APFC_BOOT_ADDR` JTAG instruction. The factory configuration image is stored in non-volatile memory and is never updated or modified using remote access. This corresponds to the default start address location `0×010000` represented in 16-bit word addressing (or the updated address if the default address is changed) in the supported parallel flash memory. For more information about the application of the `APFC_BOOT_ADDR` JTAG instruction in AP configuration scheme, refer to the "JTAG Instructions" on page 8–56.

The factory configuration image is user-designed and contains soft logic (Nios II processor or state machine and the remote communication interface) to:

- Process any errors based on status information from the dedicated remote system upgrade circuitry
- Communicate with the remote host and receive new application configurations and store the new configuration data in the local non-volatile memory device
- Determine the application configuration to be loaded into the Cyclone IV device
- Enable or disable the user watchdog timer and load its time-out value (optional)
- Instruct the dedicated remote system upgrade circuitry to start a reconfiguration cycle

Figure 8–32 shows the transitions between the factory configuration and application configuration in remote update mode.

Figure 8–32. Transitions Between Configurations in Remote Update Mode



After power up or a configuration error, the factory configuration logic writes the remote system upgrade control register to specify the address of the application configuration to be loaded. The factory configuration also specifies whether or not to enable the user watchdog timer for the application configuration and, if enabled, specifies the timer setting.

☞ Only valid application configurations designed for remote update mode include the logic to reset the timer in user mode. For more information about the user watchdog timer, refer to the "User Watchdog Timer" on page 8–78.

If there is an error while loading the application configuration, the remote system upgrade status register is written by the dedicated remote system upgrade circuitry of the Cyclone IV device to specify the cause of the reconfiguration.

The following actions cause the remote system upgrade status register to be written:

- nSTATUS driven low externally
- Internal cyclical redundancy check (CRC) error
- User watchdog timer time-out
- A configuration reset (logic array nCONFIG signal or external nCONFIG pin assertion)

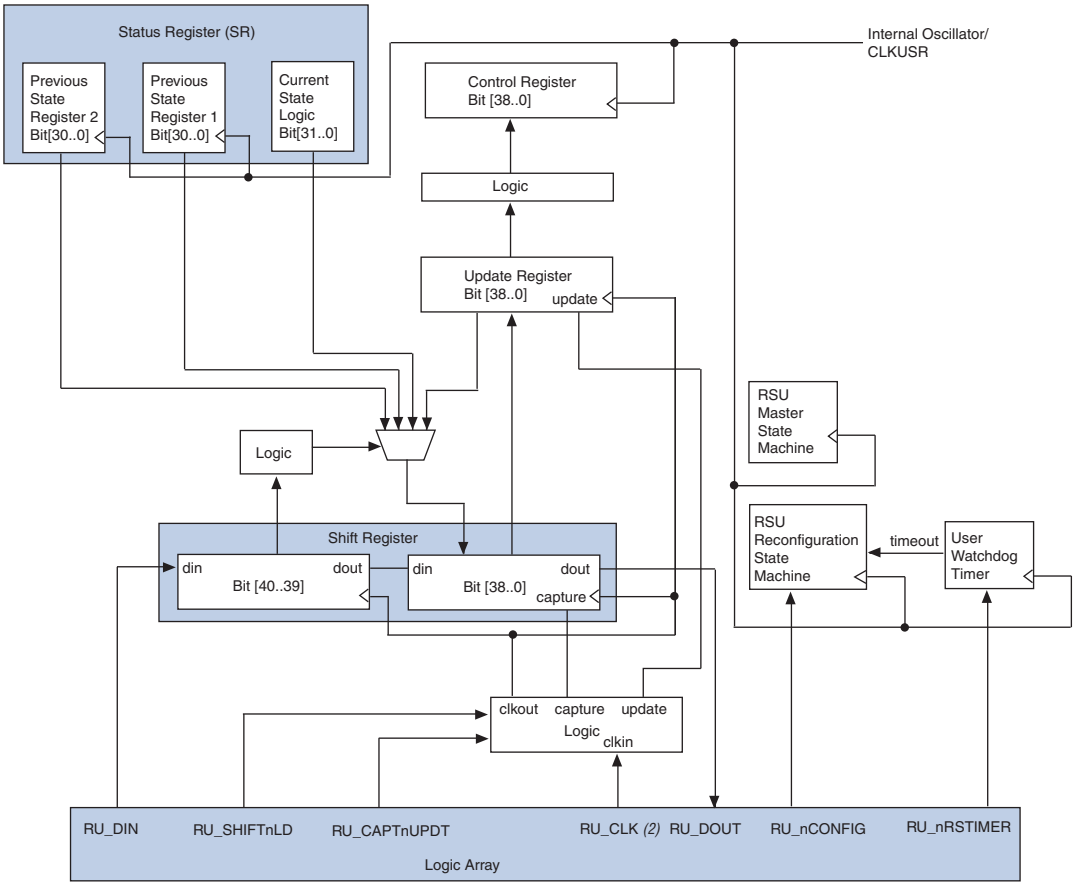
The Cyclone IV device automatically loads the factory configuration when an error occurs. This user-designed factory configuration reads the remote system upgrade status register to determine the reason for reconfiguration. Then the factory configuration takes the appropriate error recovery steps and writes to the remote system upgrade control register to determine the next application configuration to be loaded.

When Cyclone IV devices successfully load the application configuration, they enter user mode. In user mode, the soft logic (the Nios II processor or state machine and the remote communication interface) assists the Cyclone IV device in determining when a remote system update is arriving. When a remote system update arrives, the soft logic receives the incoming data, writes it to the configuration memory device and triggers the device to load the factory configuration. The factory configuration reads the remote system upgrade status register, determines the valid application configuration to load, writes the remote system upgrade control register accordingly, and starts system reconfiguration.

Dedicated Remote System Upgrade Circuitry

This section describes the implementation of the Cyclone IV device remote system upgrade dedicated circuitry. The remote system upgrade circuitry is implemented in hard logic. This dedicated circuitry interfaces with the user-defined factory application configurations implemented in the Cyclone IV device logic array to provide the complete remote configuration solution. The remote system upgrade circuitry contains the remote system upgrade registers, a watchdog timer, and state machines that control those components. Figure 8–33 shows the data path of the remote system upgrade block.

Figure 8–33. Remote System Upgrade Circuit Data Path *(Note 1)*



Notes to Figure 8–33:

(1) The RU_DOUT, RU_SHIFTnLD, RU_CAPTnUPDT, RU_CLK, RU_DIN, RU_nCONFIG, and RU_nRSTIMER signals are internally controlled by the ALTREMOTE_UPDATE megafunction.

(2) The RU_CLK refers to the ALTREMOTE_UPDATE megafunction block "clock" input. For more information, refer to the *Remote Update Circuitry (ALTREMOTE_UPDATE) Megafunction User Guide.*

Remote System Upgrade Registers

The remote system upgrade block contains a series of registers that stores the configuration addresses, watchdog timer settings, and status information. Table 8–20 lists these registers.

Table 8–20. Remote System Upgrade Registers

Register	Description
Shift register	This register is accessible by the logic array and allows the update, status, and control registers to be written and sampled by user logic. Write access is enabled in remote update mode for factory configurations to allow writing to the update register. Write access is disabled for all application configurations in remote update mode.
Control register	This register contains the current configuration address, the user watchdog timer settings, one option bit for checking early CONF_DONE, and one option bit for selecting the internal oscillator as the startup state machine clock. During a read operation in an application configuration, this register is read into the shift register. When a reconfiguration cycle is started, the contents of the update register are written into the control register.
Update register	This register contains data similar to that in the control register. However, it can only be updated by the factory configuration by shifting data into the shift register and issuing an update operation. When a reconfiguration cycle is triggered by the factory configuration, the control register is updated with the contents of the update register. During a read in a factory configuration, this register is read into the shift register.
Status register	This register is written by the remote system upgrade circuitry on every reconfiguration to record the cause of the reconfiguration. This information is used by the factory configuration to determine the appropriate action following a reconfiguration. During a capture cycle, this register is read into the shift register.

The control and status registers of the remote system upgrade are clocked by the 10-MHz internal oscillator (the same oscillator that controls the user watchdog timer) or the CLKUSR. However, the shift and update registers of the remote system upgrade are clocked by the maximum frequency of 40-MHz user clock input (RU_CLK). There is no minimum frequency for RU_CLK.

Remote System Upgrade Control Register

The remote system upgrade control register stores the application configuration address, the user watchdog timer settings, and option bits for a application configuration. In remote update mode for the AS configuration scheme, the control register address bits are set to all zeros (24'b0) at power up to load the AS factory configuration. In remote update mode for the AP configuration scheme, the control register address bits are set to 24'h010000 (24'b1 0000 0000 0000 0000) at power up to load the AP default factory configuration. However, for the AP configuration scheme, you can change the default factory configuration address to any desired address using the APFC_BOOT_ADDR JTAG instruction. Additionally, a factory configuration in remote update mode has write access to this register.

Figure 8–34 shows the control register bit positions. Table 8–21 defines the control register bit contents. The numbers in Figure 8–34 show the bit position of a setting in a register. For example, bit number 35 is the enable bit for the watchdog timer.

Figure 8–34. Remote System Upgrade Control Register

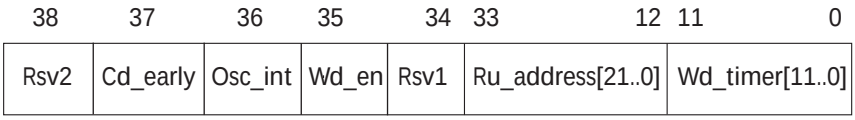


Table 8–21. Remote System Upgrade Control Register Contents

Control Register Bit	Value	Definition
Wd_timer[11..0]	12'b000000000000	User watchdog time-out value (most significant 12 bits of 29-bit count value: {Wd_timer[11..0],17'b1000})
Ru_address[21..0]	22'b0000000000000000000000	Configuration address (most significant 22 bits of 24-bit boot address value: boot_address[23:0] = {Ru_address[21..0],2'b0})
Rsv1	1'b0	Reserved bit
Wd_en	1'b1	User watchdog timer enable bit
Osc_int *(1)*	1'b1	Internal oscillator as startup state machine clock enable bit
Cd_early *(1)*	1'b1	Early CONF_DONE check
Rsv2	1'b1	Reserved bit

Note to Table 8–21:

(1) Option bit for the application configuration.

When enabled, the early CONF_DONE check (Cd_early) option bit ensures that there is a valid configuration at the boot address specified by the factory configuration and that it is of the proper size. If an invalid configuration is detected or the CONF_DONE pin is asserted too early, the device resets and then reconfigures the factory configuration image. The internal oscillator (as the startup state machine clock [Osc_int] option bit) ensures a functional startup clock to eliminate the hanging of startup. When all option bits are turned on, they provide complete coverage for the programming and startup portions of the application configuration. Altera recommends turning on both the Cd_early and Osc_int option bits.

The Cd_early and Osc_int option bits for the application configuration must be turned on by the factory configuration.

Remote System Upgrade Status Register

The remote system upgrade status register specifies the reconfiguration trigger condition. The various trigger and error conditions include:

- Cyclical redundancy check (CRC) error during application configuration
- nSTATUS assertion by an external device due to an error
- Cyclone IV device logic array triggers a reconfiguration cycle, possibly after downloading a new application configuration image
- External configuration reset (nCONFIG) assertion
- User watchdog timer time out

Table 8–22 lists the contents of the current state logic in the status register, when the remote system upgrade master state machine is in factory configuration or application configuration accessing the factory information or application information, respectively. The status register bit in Table 8–22 lists the bit positions in a 32-bit logic.

Table 8–22. Remote System Upgrade Current State Logic Contents In Status Register

Remote System Upgrade Master State Machine	Status Register Bit	Definition	Description
Factory information *(1)*	31:30	Master state machine current state	The current state of the remote system upgrade master state machine
	29:24	Reserved bits	Padding bits that are set to all 0's
	23:0	Boot address	The current 24-bit boot address that was used by the configuration scheme as the start address to load the current configuration.
Application information 1 *(2)*	31:30	Master state machine current state	The current state of the remote system upgrade master state machine
	29	User watchdog timer enable bit	The current state of the user watchdog enable, which is active high
	28:0	User watchdog timer time-out value	The current entire 29-bit watchdog time-out value.
Application information 2 *(2)*	31:30	Master state machine current state	The current state of the remote system upgrade master state machine
	29:24	Reserved bits	Padding bits that are set to all 0's
	23:0	Boot address	The current 24-bit boot address that was used as the start address to load the current configuration

Notes to Table 8–22

(1) The remote system upgrade master state machine is in factory configuration.

(2) The remote system upgrade master state machine is in application configuration.

The previous two application configurations are available in the previous state registers (previous state register 1 and previous state register 2), but only for debugging purposes.

Table 8–23 lists the contents of previous state register 1 and previous state register 2 in the status register. The status register bit in Table 8–23 shows the bit positions in a 3-bit register. The previous state register 1 and previous state register 2 have the same bit definitions. The previous state register 1 reflects the current application configuration and the previous state register 2 reflects the previous application configuration.

Table 8–23. Remote System Upgrade Previous State Register 1 and Previous State Register 2 Contents in Status Register

Status Register Bit	Definition	Description
30	nCONFIG source	One-hot, active-high field that describes the reconfiguration source that caused the Cyclone IV device to leave the previous application configuration. If there is a tie, the higher bit order indicates precedence. For example, if nCONFIG and remote system upgrade nCONFIG reach the reconfiguration state machine at the same time, the nCONFIG precedes the remote system upgrade nCONFIG.
29	CRC error source	
28	nSTATUS source	
27	User watchdog timer source	
26	Remote system upgrade nCONFIG source	
25:24	Master state machine current state	The state of the master state machine during reconfiguration causes the Cyclone IV device to leave the previous application configuration.
23:0	Boot address	The address used by the configuration scheme to load the previous application configuration.

If a capture is inappropriately done while capturing a previous state before the system has entered remote update application configuration for the first time, a value outputs from the shift register to indicate that the capture is incorrectly called.

Remote System Upgrade State Machine

The remote system upgrade control and update registers have identical bit definitions, but serve different roles (Table 8–20 on page 8–74). While both registers can only be updated when the device is loaded with a factory configuration image, the update register writes are controlled by the user logic, and the control register writes are controlled by the remote system upgrade state machine.

In factory configurations, the user logic should send the option bits (Cd_early and Osc_int), the configuration address, and watchdog timer settings for the next application configuration bit to the update register. When the logic array configuration reset (RU_nCONFIG) goes high, the remote system upgrade state machine updates the control register with the contents of the update register and starts system reconfiguration from the new application page.

To ensure the successful reconfiguration between the pages, assert the RU_nCONFIG signal for a minimum of 250 ns. This is equivalent to strobing the reconfig input of the ALTREMOTE_UPDATE megafunction high for a minimum of 250 ns.

If there is an error or reconfiguration trigger condition, the remote system upgrade state machine directs the system to load a factory or application configuration (based on mode and error condition) by setting the control register accordingly.

Table 8–24 lists the contents of the control register after such an event occurs for all possible error or trigger conditions.

The remote system upgrade status register is updated by the dedicated error monitoring circuitry after an error condition, but before the factory configuration is loaded.

Table 8–24. Control Register Contents After an Error or Reconfiguration Trigger Condition

Reconfiguration Error/Trigger	Control Register Setting In Remote Update
nCONFIG reset	All bits are 0
nSTATUS error	All bits are 0
CORE triggered reconfiguration	Update register
CRC error	All bits are 0
Wd time out	All bits are 0

User Watchdog Timer

The user watchdog timer prevents a faulty application configuration from indefinitely stalling the device. The system uses the timer to detect functional errors after an application configuration is successfully loaded into the Cyclone IV device.

The user watchdog timer is a counter that counts down from the initial value loaded into the remote system upgrade control register by the factory configuration. The counter is 29 bits wide and has a maximum count value of 2^{29}. When specifying the user watchdog timer value, specify only the most significant 12 bits. The remote system upgrade circuitry appends 17'b1000 to form the 29-bits value for the watchdog timer. The granularity of the timer setting is 2^{17} cycles. The cycle time is based on the frequency of the 10-MHz internal oscillator or CLKUSR (maximum frequency of 40 MHz).

Table 8–25 lists the operating range of the 10-MHz internal oscillator.

Table 8–25. 10-MHz Internal Oscillator Specifications

Minimum	Typical	Maximum	Unit
5	6.5	10	MHz

The user watchdog timer begins counting after the application configuration enters device user mode. This timer must be periodically reloaded or reset by the application configuration before the timer expires by asserting `RU_nRSTIMER`. If the application configuration does not reload the user watchdog timer before the count expires, a time-out signal is generated by the remote system upgrade dedicated circuitry. The time-out signal tells the remote system upgrade circuitry to set the user watchdog timer status bit (`Wd`) in the remote system upgrade status register and reconfigures the device by loading the factory configuration.

To allow the remote system upgrade dedicated circuitry to reset the watchdog timer, you must assert the `RU_nRSTIMER` signal active for a minimum of 250 ns. This is equivalent to strobing the `reset_timer` input of the ALTREMOTE_UPDATE megafunction high for a minimum of 250 ns.

Errors during configuration are detected by the CRC engine. Functional errors must not exist in the factory configuration because it is stored and validated during production and is never updated remotely.

The user watchdog timer is disabled in factory configurations and during the configuration cycle of the application configuration. It is enabled after the application configuration enters user mode.

Quartus II Software Support

Implementation in your design requires a remote system upgrade interface between the Cyclone IV device logic array and remote system upgrade circuitry. You must also generate configuration files for production and remote programming of the system configuration memory. The Quartus II software provides these features.

The two implementation options, ALTREMOTE_UPDATE megafunction and remote system upgrade atom, are for the interface between the remote system upgrade circuitry and the device logic array interface. Using the megafunction block instead of creating your own logic saves design time and offers more efficient logic synthesis and device implementation.

For more information about the ALTREMOTE_UPDATE megafunction, refer to the *Remote Update Circuitry (ALTREMOTE_UPDATE) Megafunction User Guide*.

Document Revision History

Table 8–26 lists the revision history for this chapter.

Table 8–26. Document Revision History

Date	Version	Changes Made
December 2010	1.3	■ Updated for the Quartus II software version 10.1 release. ■ Added Cyclone IV E new device package information. ■ Updated Table 8–7, Table 8–10, and Table 8–11. ■ Minor text edits.
July 2010	1.2	Updated for the Quartus II software 10.0 release: ■ Updated "Power-On Reset (POR) Circuit", "Configuration and JTAG Pin I/O Requirements", and "Reset" sections. ■ Updated Figure 8–10. ■ Updated Table 8–16 and Table 8–17.
February 2010	1.1	Updated for the Quartus II software 9.1 SP1 release: ■ Added"Overriding the Internal Oscillator" and "AP Configuration (Supported Flash Memories)" sections. ■ Updated "JTAG Instructions" section. ■ Added Table 8–6. ■ Updated Table 8–2, Table 8–3, Table 8–4, Table 8–6, Table 8–11, Table 8–13, Table 8–14, Table 8–15, and Table 8–18. ■ Updated Figure 8–4, Figure 8–5, Figure 8–6, Figure 8–13, Figure 8–14, Figure 8–15, Figure 8–17, Figure 8–18, Figure 8–23, Figure 8–24, Figure 8–25, Figure 8–26, Figure 8–27, Figure 8–28, and Figure 8–29.
November 2009	1.0	Initial release.

9. Configuration, Design Security, and Remote System Upgrades in the Cyclone III Device Family

CIII51016-1.2

This chapter describes the configuration, design security, and remote system upgrades in Cyclone® III devices. The Cyclone III device family (Cyclone III and Cyclone III LS devices) uses SRAM cells to store configuration data. Configuration data must be downloaded to Cyclone III device family each time the device powers up because SRAM memory is volatile.

The Cyclone III device family is configured using one of the following configuration schemes:

- Fast Active serial (AS)
- Active parallel (AP) for Cyclone III devices only
- Passive serial (PS)
- Fast passive parallel (FPP)
- Joint Test Action Group (JTAG)

All configuration schemes use an external configuration controller (for example, MAX® II devices or a microprocessor), a configuration device, or a download cable.

The Cyclone III device family offers the following configuration features:

- Configuration data decompression
- Design security (for Cyclone III LS devices only)
- Remote system upgrade

As Cyclone III LS devices play a role in larger and more critical designs in competitive commercial and military environments, it is increasingly important to protect your designs from copying, reverse engineering, and tampering. Cyclone III LS devices address these concerns with 256-bit advanced encryption standard (AES) programming file encryption and anti-tamper feature support to prevent tampering. For more information about the design security feature in Cyclone III LS devices, refer to "Design Security" on page 9–73.

System designers face difficult challenges such as shortened design cycles, evolving standards, and system deployments in remote locations. The Cyclone III device family helps overcome these challenges with inherent re-programmability and dedicated circuitry to perform remote system upgrades. Remote system upgrades help deliver feature enhancements and bug fixes without costly recalls, reduce time-to-market, and extend product life. Remote system upgrades can also be implemented with the advanced Cyclone III device family features such as real-time decompression of configuration data. For more information about the remote system upgrade feature in Cyclone III device family, refer to "Remote System Upgrade" on page 9–77.

This chapter describes the Cyclone III device family configuration features and describes how to configure Cyclone III device family using the supported configuration schemes. This chapter also includes configuration pin descriptions and the Cyclone III device family configuration file formats. In this chapter, the generic term "device" includes all Cyclone III device family.

This chapter contains the following sections:

- "Configuration Features" on page 9–2
- "Design Security" on page 9–73
- "Remote System Upgrade" on page 9–77

Configuration Features

Cyclone III device family offers configuration data decompression to reduce configuration file storage, provides design security feature to protect your configuration data (for Cyclone III LS devices only), and provides remote system upgrade to allow you to remotely update your Cyclone III device family designs.

Table 9–1 lists which configuration methods you can use in each configuration scheme.

Table 9–1. Cyclone III Device Family Configuration Features

Configuration Scheme	Configuration Method	Decompression	Remote System Upgrade *(1)*	Design Security (Cyclone III LS Devices Only)
Fast Active Serial Standard (AS Standard POR)	Serial Configuration Device	✓	✓	✓
Fast Active Serial Fast (AS Fast POR)	Serial Configuration Device	✓	✓	✓
Active Parallel ×16 Standard (AP Standard POR, for Cyclone III devices only)	Supported Flash Memory *(2)*	—	✓	—
Active Parallel ×16 Fast (AP Fast POR, for Cyclone III devices only)	Supported Flash Memory *(2)*	—	✓	—
Passive Serial Standard (PS Standard POR)	External Host with Flash Memory	✓	—	✓
	Download Cable	✓	—	✓ *(3)*
Passive Serial Fast (PS Fast POR)	External Host with Flash Memory	✓	—	✓
	Download Cable	✓	—	✓ *(3)*
Fast Passive Parallel Fast (FPP Fast POR)	External Host with Flash Memory	—	—	✓
JTAG based configuration	External Host with Flash Memory	—	—	—
	Download Cable	—	—	—

Notes to Table 9–1:

(1) Remote update mode is supported when using the remote system upgrade feature. You can enable or disable remote update mode with an option setting in the Quartus® II software. For more information about the remote system upgrade feature, refer to "Remote System Upgrade" on page 9–77.

(2) For more information about the supported families for the Numonyx commodity parallel flash, refer to Table 9–11 on page 9–24.

(3) The design security feature is not supported using a SRAM Object File (**.sof**).

The design security feature is for Cyclone III LS devices only and is available in all configuration schemes except the JTAG-based configuration. The decompression feature is not supported when you have enabled the design security feature.

When using a serial configuration scheme such as PS or fast AS, the configuration time is the same whether or not you have enabled the design security feature. A ×4 DCLK is required if you use the FPP scheme with the design security feature.

Cyclone III devices support remote system upgrade in AS and AP configuration schemes. Cyclone III LS devices only support remote system upgrade in the AS configuration scheme.

This section only describes the decompression feature. For more information about the design security and remote system upgrade, refer to "Design Security" on page 9–73 and "Remote System Upgrade" on page 9–77.

Configuration Data Decompression

Cyclone III device family supports configuration data decompression, which saves configuration memory space and time. This feature allows you to store compressed configuration data in configuration devices or other memory and send the compressed bitstream to Cyclone III device family. During configuration, Cyclone III device family decompress the bitstream in real time and program SRAM cells. The decompression feature is not supported when you have enabled the design security feature.

Preliminary data indicates that compression reduces configuration bitstream size by 35 to 55%.

Cyclone III device family supports decompression in the AS and PS configuration schemes. Decompression is not supported in the AP, FPP, or JTAG-based configuration schemes. In PS mode, use the Cyclone III device family decompression feature to reduce configuration time.

Altera recommends using the Cyclone III device family decompression feature during AS configuration if you must save configuration memory space in the serial configuration device.

When you enable compression, the Quartus II software generates configuration files with compressed configuration data. This compressed file reduces the storage requirements in the configuration device or flash memory and decreases the time needed to send the bitstream to the Cyclone III device family. The time needed by a Cyclone III device family to decompress a configuration file is less than the time needed to send the configuration data to the device. There are two methods for enabling compression for Cyclone III device family bitstreams in the Quartus II software:

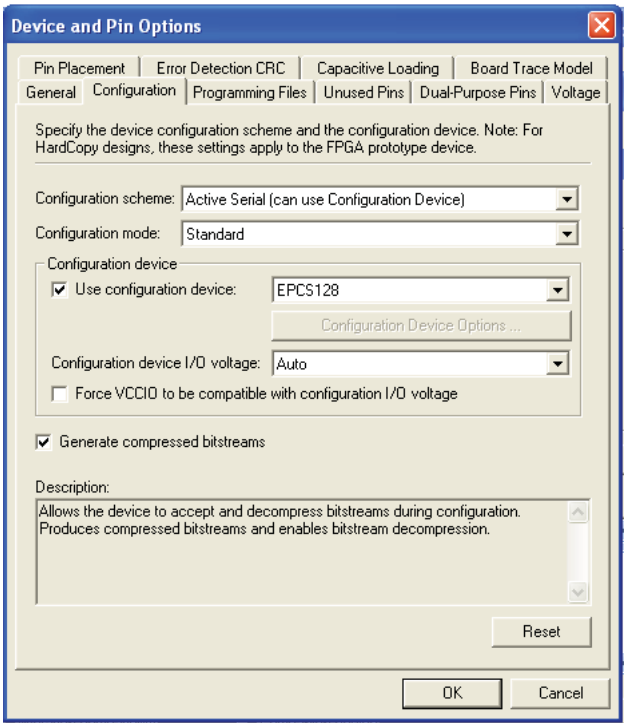
- Before design compilation (using the Compiler Settings menu).
- After design compilation (use the **Convert Programming Files** dialog box).

To enable compression in the compiler settings of the project in the Quartus II software, perform the following steps:

1. On the Assignments menu, click **Device**. The **Settings** dialog box appears.
2. Click **Device and Pin Options**. The **Device and Pin Options** dialog box appears.
3. Click the **Configuration** tab.

4. Turn on **Generate compressed bitstreams** (Figure 9–1).
5. Click **OK**.
6. In the **Settings** dialog box, click **OK**.

Figure 9–1. Enabling Compression for Cyclone III Device Family Bitstreams in Compiler Settings

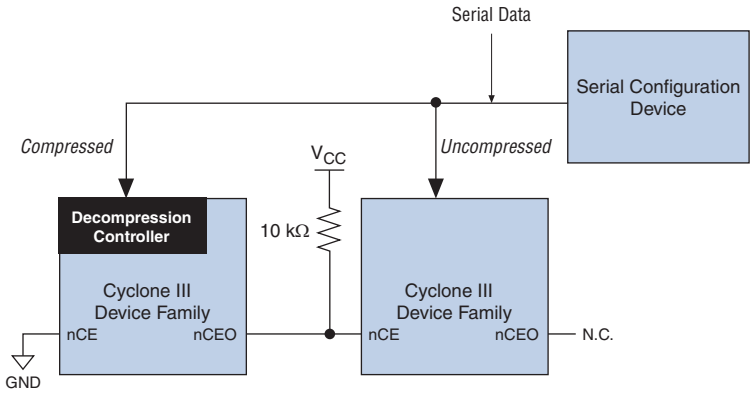


To enable compression when creating programming files from the **Convert Programming Files** window, follow these steps:

1. On the File menu, click **Convert Programming Files**.
2. Under **Output programming file**, from the pull-down menu, select your desired file type.
3. If you select the Programmer Object File (**.pof**), you must specify a configuration device, directly under the file type.
4. In the **Input files to convert** box, select **SOF Data**.
5. Click **Add File** to browse to the Cyclone III device family **.sof**s.
6. In the **Convert Programming Files** dialog box, select the **.pof** you added to **SOF Data** and click **Properties**.
7. In the **SOF File Properties** dialog box, turn on the **Compression** option.

When multiple devices in Cyclone III device family are cascaded, you can selectively enable the compression feature for each device in the chain. Figure 9–2 shows a chain of two devices in Cyclone III device family. The first device has compression enabled and receives compressed bitstream from the configuration device. The second device has the compression feature disabled and receives uncompressed data. You can generate programming files for this setup from the **Convert Programming Files** dialog box from the File menu in the Quartus II software.

Figure 9–2. Compressed and Uncompressed Configuration Data in the Same Configuration File



Configuration Requirement

The following section describes power-on-reset (POR) for Cyclone III device family.

POR Circuit

The POR circuit keeps the device in the reset state until the power supply voltage levels have stabilized after device power-up. After device power-up, the device does not release nSTATUS until the required voltages listed in table Table 9–4 on page 9–8 are above the POR trip point of the device. V_{CCINT} and V_{CCA} are monitored for brown-out conditions after device power-up.

V_{CCA} is the analog power to the phase-locked loop (PLL).

In Cyclone III device family, you can select either a fast POR time or standard POR time depending on the MSEL pin settings. The fast POR time is 3 ms < TPOR < 9 ms for the fast configuration time. The standard POR time is 50 ms < TPOR < 200 ms, which has a lower power-ramp rate.

Table 9–2 lists the supported POR times for each configuration scheme.

Table 9–2. Cyclone III Device Family Supported POR Times Across Configuration Schemes *(Note 1)* (Part 1 of 2)

Configuration Scheme	Fast POR Time (3 ms< TPOR < 9 ms)	Standard POR Time (50 ms< TPOR < 200 ms)	Configuration Voltage Standard (V)*(2)*
Fast Active Serial Standard (AS Standard POR)	—	✓	3.3
Fast Active Serial Standard (AS Standard POR)	—	✓	3.0/2.5
Fast Active Serial Fast (AS Fast POR)	✓	—	3.3
Fast Active Serial Fast (AS Fast POR)	✓	—	3.0/2.5

Table 9–2. Cyclone III Device Family Supported POR Times Across Configuration Schemes *(Note 1)* (Part 2 of 2)

Configuration Scheme	Fast POR Time (3 ms< TPOR < 9 ms)	Standard POR Time (50 ms< TPOR < 200 ms)	Configuration Voltage Standard (V)*(2)*
Active Parallel ×16 Standard (AP Standard POR, for Cyclone III devices only)	—	✓	3.3
Active Parallel ×16 Standard (AP Standard POR, for Cyclone III devices only)	—	✓	3.0/2.5
Active Parallel ×16 Standard (AP Standard POR, for Cyclone III devices only)	—	✓	1.8
Active Parallel ×16 Fast (AP Fast POR, for Cyclone III devices only)	✓	—	3.3
Active Parallel ×16 Fast (AP Fast POR, for Cyclone III devices only)	✓	—	1.8
Passive Serial Standard (PS Standard POR)	—	✓	3.3/3.0/2.5
Passive Serial Fast (PS Fast POR)	✓	—	3.3/3.0/2.5
Fast Passive Parallel Fast (FPP Fast POR)	✓	—	3.3/3.0/2.5
Fast Passive Parallel Fast (FPP Fast POR)	✓	—	1.8/1.5
JTAG-based configuration	*(3)*	*(3)*	—

Notes to Table 9–2:

(1) Altera recommends connecting the MSEL pins to V_{CCA} or GND depending on the MSEL pin settings.

(2) The configuration voltage standard is applied to the V_{CCIO} supply of the bank in which the configuration pins reside.

(3) JTAG-based configuration takes precedence over other configuration schemes, which means the MSEL pin settings are ignored. However, the POR time is dependent on the MSEL pin settings.

In some applications, it is necessary for a device to wake up very quickly to begin operation. The Cyclone III device family offers the fast **POR time** option to support fast wake-up time applications. The **fast POR time** option has stricter power-up requirements when compared with the **standard POR time** option. You can select either the fast POR or standard POR options with the MSEL pin settings.

The automotive application is for Cyclone III devices only. The Cyclone III devices fast wake-up time meets the requirement of common bus standards in automotive applications, such as Media Orientated Systems Transport (MOST) and Controller Area Network (CAN).

For more information about wake-up time and the POR circuit, refer to the *Hot-Socketing and Power-On Reset in Cyclone III Devices* chapter.

Configuration File Size

Table 9–3 lists the approximate uncompressed configuration file sizes for Cyclone III device family. To calculate the amount of storage space required for multiple device configurations, add the file size of each device together.

Table 9–3. Cyclone III Device Family Uncompressed Raw Binary File (.rbf) Sizes

Device		Data Size (bits)
Cyclone III	EP3C5	3,000,000
	EP3C10	3,000,000
	EP3C16	4,100,000
	EP3C25	5,800,000
	EP3C40	9,600,000
	EP3C55	14,900,000
	EP3C80	20,000,000
	EP3C120	28,600,000
Cyclone III LS	EP3CLS70	25,165,824 *(1)*
	EP3CLS100	25,165,824 *(1)*
	EP3CLS150	50,331,648 *(1)*
	EP3CLS200	50,331,648 *(1)*

Note to Table 9–3:

(1) These values are preliminary.

Use the data in Table 9–3 only to estimate the file size before design compilation. Different configuration file formats, such as Hexadecimal (**.hex**) or Tabular Text File (**.ttf**) formats, have different file sizes. However, for any specific version of the Quartus II software, any design targeted for the same device has the same uncompressed configuration file size. If you are using compression, the file size varies after each compilation because the compression ratio is design dependent.

For more information about setting device configuration options or creating configuration files, refer to the *Software Settings* section in volume 2 of the *Configuration Handbook*.

Configuration and JTAG Pin I/O Requirements

Cyclone III devices are manufactured using the TSMC 65-nm low-k dielectric process; Cyclone III LS devices are manufactured using the TSMC 60-nm low-k dielectric process. Although Cyclone III device family uses TSMC 2.5-V transistor technology in the I/O buffers, the devices are compatible and able to interface with 2.5-, 3.0-, 3.3-V configuration voltage standards. However, you must follow specific requirements when interfacing Cyclone III device family with 2.5-, 3.0-, 3.3-V configuration voltage standards.

All I/O inputs must maintain a maximum AC voltage of 4.1 V. When using a JTAG configuration scheme or a serial configuration device in an AS configuration scheme, you must connect a 25-Ω series resistor at the near end of the `TDO` and `TDI` pin or the serial configuration device for the `DATA[0]` pin. When cascading Cyclone III device family in a multi-device configuration, you must connect the repeater buffers between the master and slave devices for `DATA` and `DCLK`.

The output resistance of the repeater buffers must fit the maximum overshoot equation shown in Equation 9–1:

Equation 9–1. *(Note 1)*

$$0.8Z_O \leq R_E \leq 1.8Z_O$$

Note to Equation 9–1:

(1) Z_0 is the transmission line impedance and R_E is the equivalent resistance of the output buffer.

Configuration Process

This section describes the configuration process.

For more information about the configuration cycle state machine of Altera® FPGAs, refer to the *Configuring Altera FPGAs* chapter in volume 1 of the *Configuration Handbook*.

Power Up

If the device is powered up from the power-down state, the V_{CCIO} for all the I/O banks must be powered up to the appropriate level for the device to exit POR.

To begin configuration, the required voltages listed in Table 9–4 must be powered up to the appropriate voltage levels.

Table 9–4. Power-Up Voltage for Cyclone III Device Family Configuration

Device	Voltage that must be Powered-Up *(1)*
Cyclone III	V_{CCINT}, V_{CCA}, V_{CCIO} *(2)*
Cyclone III LS	V_{CCBAT}, V_{CCINT}, V_{CCA}, V_{CCIO} *(2)*

Notes to Table 9–4:

(1) Voltages must be powered up to the appropriate voltage levels to begin configuration.

(2) V_{CCIO} is for banks in which the configuration and JTAG pins reside.

Reset

When nCONFIG or nSTATUS is low, the device is in reset. Upon power-up, the Cyclone III device family goes through POR. POR delay is dependent on the MSEL pin settings, which correspond to the configuration scheme that you selected. Depending on the configuration scheme, either a fast or standard POR time is available. The fast POR time is 3 ms < TPOR < 9 ms for a fast configuration time. The standard POR time is 50 ms < TPOR < 200 ms, which has a lower power-ramp rate.

During POR, the device resets, holds nSTATUS and CONF_DONE low, and tri-states all user I/O pins. When the device exits POR, all user I/O pins continue to tri-state. The user I/O pins and dual-purpose I/O pins have weak pull-up resistors that are always enabled (after POR) before and during configuration. After POR, the Cyclone III device family releases nSTATUS, which is pulled high by an external 10-kΩ pull-up resistor and enters configuration mode.

When nCONFIG goes high, the device exits reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. After nSTATUS is released, the device is ready to receive configuration data and the configuration stage begins.

Cyclone III LS devices are accessible by limited JTAG instructions after POR. For more information about enabling full JTAG instructions access, refer to "JTAG Instructions" on page 9–61.

For more information about the value of weak pull-up resistors on the I/O pins that are on before and during configuration, refer to the *Cyclone III Device Data Sheet* and *Cyclone III LS Device Data Sheet* chapters.

Configuration

Configuration data is latched into the Cyclone III device family at each DCLK cycle. However, the width of the data bus and the configuration time taken for each scheme are different. After the device receives all the configuration data, the device releases the open-drain CONF_DONE pin, which is pulled high by an external 10-kΩ pull-up resistor. A low-to-high transition on the CONF_DONE pin indicates that configuration is complete and initialization of the device can begin. The CONF_DONE pin must have an external 10-kΩ pull-up resistor for the device to initialize.

You can begin reconfiguration by pulling the nCONFIG pin low. The nCONFIG pin must be low for at least 500 ns. When nCONFIG is pulled low, the Cyclone III device family is reset. The Cyclone III device family also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. When nCONFIG returns to a logic-high level and nSTATUS is released by the Cyclone III device family, reconfiguration begins.

Configuration Error

If an error occurs during configuration, the Cyclone III device family asserts the nSTATUS signal low, indicating a data frame error, and the CONF_DONE signal stays low. If the **Auto-restart configuration after error** option (available in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box) is turned on, the Cyclone III device family releases nSTATUS after a reset time-out period (a maximum of 230 µs), and retries configuration. If this option is turned off, the system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 500 ns to restart configuration.

Initialization

In Cyclone III device family, the clock source for initialization is either a 10-MHz (typical) internal oscillator (separate from the AS internal oscillator) or an optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If you use the internal oscillator, the device provides itself with enough clock cycles for a proper initialization. When using the internal oscillator, you do not need to send additional clock cycles from an external source to the CLKUSR pin during the initialization stage. Additionally, you can use the CLKUSR pin as a user I/O pin.

You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the **CLKUSR** option. The CLKUSR pin allows you to control when your device enters user mode for an indefinite amount of time. You can turn on the **Enable user-supplied start-up clock (CLKUSR)** option in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box. When you turn on

the **Enable user supplied start-up clock option (CLKUSR)** option, the CLKUSR pin is the initialization clock source. Supplying a clock on the CLKUSR pin does not affect the configuration process. After the configuration data is accepted and CONF_DONE goes high, the Cyclone III device family requires a certain amount of clock cycles to initialize and to enter user mode.

Table 9–5 lists the required clock cycles for proper initialization in Cyclone III device family.

Table 9–5. Initialization Clock Cycles Required in Cyclone III Device Family

Device	Initialization Clock Cycles
Cyclone III	3,185
Cyclone III LS	3,192

Table 9–6 lists the maximum CLKUSR frequency (f_{MAX}) for Cyclone III device family.

Table 9–6. Maximum CLKUSR Frequency for Cyclone III Device Family

Device	f_{MAX} (MHz)
Cyclone III	133
Cyclone III LS	100

If you use the optional CLKUSR pin and the nCONFIG pin is pulled low to restart configuration during device initialization, ensure that the CLKUSR pin continues to toggle when nSTATUS is low (a maximum of 230 µs).

User Mode

An optional INIT_DONE pin is available that signals the end of initialization and the start of user mode with a low-to-high transition. The **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box. If you use the INIT_DONE pin, it is high due to an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. After the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. This low-to-high transition signals that the device has entered user mode. In user mode, the user I/O pins function as assigned in your design and no longer have weak pull-up resistors.

Configuration Scheme

A configuration scheme with different configuration voltage standards is selected by driving the MSEL pins either high or low, as listed in Table 9–7.

The MSEL pins are powered by V_{CCINT}. The MSEL[3..0] pins have 9-kΩ internal pull-down resistors that are always active.

☞ Smaller Cyclone III devices or package options (E144, M164, Q240, F256, and U256 packages) do not have the `MSEL[3]` pin. The AS Fast POR configuration scheme at 3.0- or 2.5-V configuration voltage standard and the AP configuration scheme are not supported in Cyclone III devices without the `MSEL[3]` pin. To configure these devices with other supported configuration schemes, select the `MSEL[2..0]` pins according to the MSEL settings in Table 9–7.

☞ Hardwire the MSEL pins to V_{CCA} or GND without any pull-up or pull-down resistors to avoid any problems detecting an incorrect configuration scheme. Do not drive the MSEL pins with a microprocessor or another device.

☞ The Quartus II software prohibits you from using the LVDS I/O standard in I/O Bank 1 when the configuration device I/O voltage is not 2.5 V. If you need to assign LVDS I/O standard in I/O Bank 1, navigate to Assignments>Device>Settings>Device and Pin Option>Configuration to change the Configuration Device I/O voltage to **2.5 V** or **Auto**.

Table 9–7. Cyclone III Device Family Configuration Schemes *(Note 1)* (Part 1 of 2)

Configuration Scheme	MSEL				Configuration Voltage Standard (V) *(2)*,*(3)*
	3	2	1	0	
Fast Active Serial Standard (AS Standard POR)	0	0	1	0	3.3
Fast Active Serial Standard (AS Standard POR)	0	0	1	1	3.0/2.5
Fast Active Serial Fast (AS Fast POR)	1	1	0	1	3.3
Fast Active Serial Fast (AS Fast POR)	0	1	0	0	3.0/2.5
Active Parallel ×16 Standard (AP Standard POR, for Cyclone III devices only)	0	1	1	1	3.3
Active Parallel ×16 Standard (AP Standard POR, for Cyclone III devices only)	1	0	1	1	3.0/2.5
Active Parallel ×16 Standard (AP Standard POR, for Cyclone III devices only)	1	0	0	0	1.8
Active Parallel ×16 Fast (AP Fast POR, for Cyclone III devices only)	0	1	0	1	3.3
Active Parallel ×16 Fast (AP Fast POR, for Cyclone III devices only)	0	1	1	0	1.8
Passive Serial Standard (PS Standard POR)	0	0	0	0	3.3/3.0/2.5
Passive Serial Fast (PS Fast POR)	1	1	0	0	3.3/3.0/2.5
Fast Passive Parallel Fast (FPP Fast POR) *(4)*	1	1	1	0	3.3/3.0/2.5
Fast Passive Parallel Fast (FPP Fast POR) (for Cyclone III devices only)*(4)*	1	1	1	1	1.8/1.5
Fast Passive Parallel Fast (FPP Fast POR) (for Cyclone III LS devices only)	0	0	0	1	1.8/1.5
Fast Passive Parallel Fast (FPP Fast POR) with Encryption (for Cyclone III LS devices only)	0	1	0	1	3.3/3.0/2.5

Table 9–7. Cyclone III Device Family Configuration Schemes *(Note 1)* (Part 2 of 2)

Configuration Scheme	MSEL				Configuration Voltage Standard (V) *(2)*,*(3)*
	3	2	1	0	
Fast Passive Parallel Fast (FPP Fast POR) with Encryption (for Cyclone III LS devices only)	0	1	1	0	1.8/1.5
JTAG-based configuration *(5)*	*(6)*	*(6)*	*(6)*	*(6)*	—

Notes to Table 9–7:

(1) Altera recommends connecting the MSEL pins to V_{CCA} or GND depending on the MSEL pin settings.

(2) The configuration voltage standard is applied to the V_{CCIO} supply of the bank in which the configuration pins reside.

(3) You must follow specific requirements when interfacing Cyclone III device family with 2.5-, 3.0-, and 3.3-V configuration voltage standards. For more information about these requirements, refer to "Configuration and JTAG Pin I/O Requirements" on page 9–7.

(4) FPP configuration is not supported in the Cyclone III E144 device package of Cyclone III devices.

(5) The JTAG-based configuration takes precedence over other configuration schemes, which means the MSEL pin settings are ignored.

(6) Do not leave the MSEL pins floating. Connect them to V_{CCA} or GND. These pins support the non-JTAG configuration scheme used in production. Altera recommends connecting the MSEL pins to GND if your device is only using the JTAG configuration.

AS Configuration (Serial Configuration Devices)

In the AS configuration scheme, Cyclone III device family is configured using a serial configuration device. These configuration devices are low-cost devices with non-volatile memories that feature a simple four-pin interface and a small form factor. These features make serial configuration devices the ideal low-cost configuration solution.

For more information about serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

In Cyclone III device family, the active master clock frequency runs at a maximum of 40 MHz, and typically at 30 MHz. Cyclone III device family only work with serial configuration devices that support up to 40 MHz.

Serial configuration devices provide a serial interface to access configuration data. During device configuration, Cyclone III device family reads configuration data using the serial interface, decompress data if necessary, and configure their SRAM cells. This scheme is referred to as the AS configuration scheme because the device controls the configuration interface.

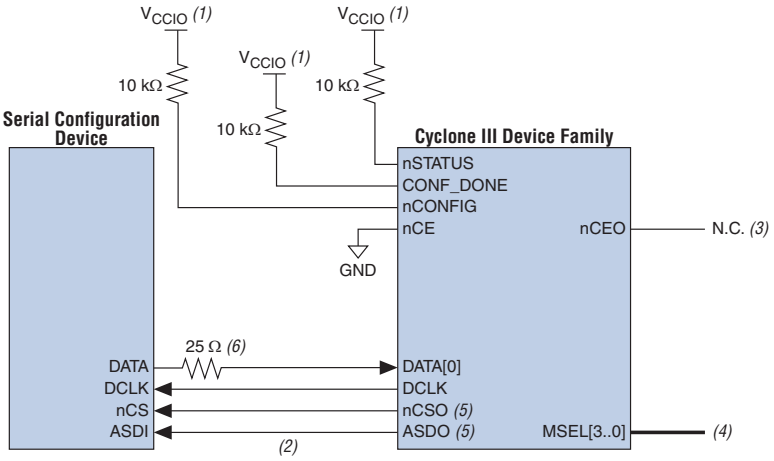
Single-Device AS Configuration

The four-pin interface of serial configuration devices consists of the following pins:

- Serial clock input (DCLK)
- Serial data output (DATA)
- AS data input (ASDI)
- Active-low chip select (nCS)

This four-pin interface connects to Cyclone III device family pins, as shown in Figure 9–3.

Figure 9–3. Single-Device AS Configuration



Notes to Figure 9–3:

(1) Connect the pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) Cyclone III device family uses the ASDO-to-ASDI path to control the configuration device.

(3) The nCEO pin is left unconnected or used as a user I/O pin when it does not feed the nCE pin of another device.

(4) The MSEL pin settings vary for different configuration voltage standards and POR time. To connect MSEL[3..0], refer to Table 9–7 on page 9–11. Connect the MSEL pins directly to V_{CCA} or GND.

(5) These are dual-purpose I/O pins. The nCSO pin functions as the FLASH_NCE pin in AP mode. The ASDO pin functions as the DATA[1] pin in other AP and FPP modes.

(6) Connect the series resistor at the near end of the serial configuration device.

When connecting a serial configuration device to a Cyclone III device family in the single-device AS configuration, you must connect a 25-Ω series resistor at the near end of the serial configuration device for DATA[0]. The 25-Ω resistor in the series works to minimize the driver impedance mismatch with the board trace and reduce overshoot seen at the Cyclone III device family DATA[0] input pin.

In a single-device AS configuration, the maximum board loading and board trace length between the supported serial configuration device and the Cyclone III device family must follow the recommendations in Table 9–9 on page 9–20.

The DCLK generated by the Cyclone III device family controls the entire configuration cycle and provides timing for the serial interface. Cyclone III device family uses a 40-MHz internal oscillator to generate DCLK. There are some variations in the internal oscillator frequency because of the process, voltage, and temperature conditions in Cyclone III device family. The internal oscillator is designed to ensure that its maximum frequency is guaranteed to meet the EPCS device specifications.

EPCS1 does not support Cyclone III device family because of its insufficient memory capacity.

Table 9–8 lists the AS DCLK output frequency for Cyclone III device family.

Table 9–8. AS DCLK Output Frequency

Oscillator	Minimum	Typical	Maximum	Unit
40 MHz	20	30	40	MHz

In the AS configuration scheme, the serial configuration device latches input and control signals on the rising edge of DCLK and drives out configuration data on the falling edge. Cyclone III device family drives out control signals on the falling edge of DCLK and latch configuration data on the falling edge of DCLK.

In configuration mode, the Cyclone III device family enables the serial configuration device by driving the nCSO output pin low, which connects to the nCS pin of the configuration device. The Cyclone III device family uses the DCLK and DATA[1] pins to send operation commands and read address signals to the serial configuration device. The configuration device provides data on its DATA pin, which connects to the DATA[0] input of the Cyclone III device family.

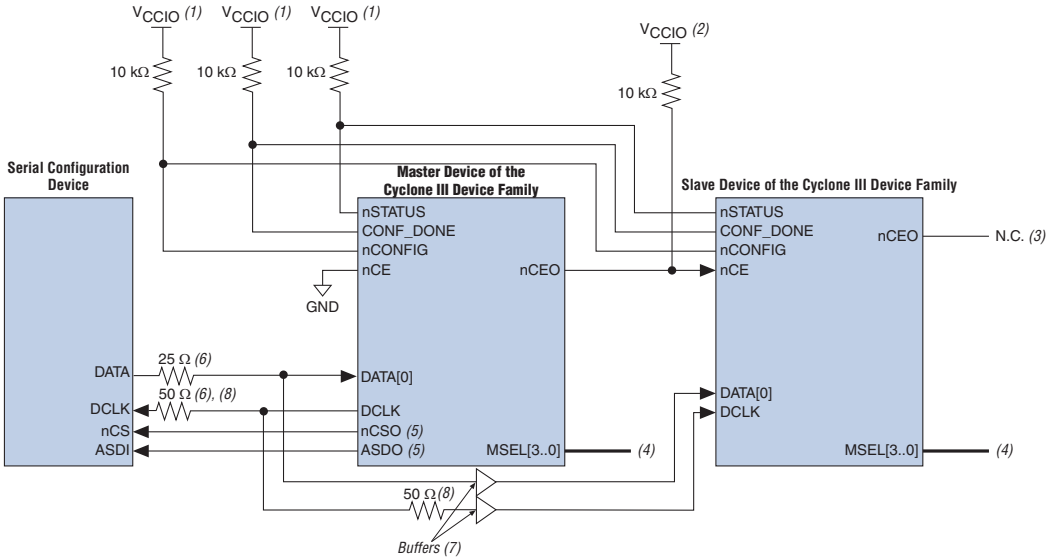
After all the configuration bits are received by the Cyclone III device family, it releases the open-drain CONF_DONE pin, which is pulled high by an external 10-kΩ resistor. Initialization begins only after the CONF_DONE signal reaches a logic-high level. All AS configuration pins (DATA[0], DCLK, nCSO, and DATA[1]) have weak internal pull-up resistors that are always active. After configuration, these pins are set as input tri-stated and are driven high by weak internal pull-up resistors. The CONF_DONE pin must have an external 10-kΩ pull-up resistor for the device to initialize.

The timing parameters for AS mode are not listed here because the t_{CF2CD}, t_{CF2ST0}, t_{CFG}, t_{STATUS}, t_{CF2ST1}, and t_{CD2UM} timing parameters are identical to the timing parameters for PS mode listed in Table 9–13 on page 9–39.

Multi-Device AS Configuration

You can configure multiple Cyclone III device family using a single serial configuration device. You can cascade multiple Cyclone III device family using the chip-enable (nCE) and chip-enable-out (nCEO) pins. The first device in the chain must have its nCE pin connected to GND. You must connect its nCEO pin to the nCE pin of the next device in the chain. Use an external 10-kΩ pull-up resistor to pull the nCEO signal high to its V_{CCIO} level to help the internal weak pull-up resistor. When the first device captures all its configuration data from the bitstream, it drives the nCEO pin low, enabling the next device in the chain. You can leave the nCEO pin of the last device unconnected or use it as a user I/O pin after configuration if the last device in the chain is a Cyclone III device family. The nCONFIG, nSTATUS, CONF_DONE, DCLK, and DATA[0] pins of each device in the chain are connected (Figure 9–4).

Figure 9–4. Multi-device AS Configuration



Notes to Figure 9–4:

(1) Connect the pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) Connect the pull-up resistor to the V_{CCIO} supply voltage of the I/O bank in which the nCE pin resides.

(3) You can leave the nCEO pin unconnected or use it as a user I/O pin when it does not feed the nCE pin of another device.

(4) The MSEL pin settings vary for different configuration voltage standards and POR time. You must set the master device of the Cyclone III device family in AS mode and the slave devices in PS mode. To connect MSEL[3..0] for the master device in AS mode and slave devices in PS mode, refer to Table 9–7 on page 9–11. Connect the MSEL pins directly to V_{CCA} or GND.

(5) These are dual-purpose I/O pins. The nCSO pin functions as the FLASH_NCE pin in AP mode. The ASDO pin functions as the DATA[1] pin in other AP and FPP modes.

(6) Connect the series resistor at the near end of the serial configuration device.

(7) Connect the repeater buffers between the master and slave devices of the Cyclone III device family for DATA[0] and DCLK. All I/O inputs must maintain a maximum AC voltage of 4.1 V. The output resistance of the repeater buffers must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 9–7.

(8) The 50-Ω series resistors are optional if the 3.3-V configuration voltage standard is applied. For optimal signal integrity, connect these 50-Ω series resistors if the 2.5- or 3.0-V configuration voltage standard is applied.

The first Cyclone III device family in the chain is the configuration master and controls the configuration of the entire chain. You must connect its MSEL pins to select the AS configuration scheme. The remaining Cyclone III device family is configuration slaves and you must connect their MSEL pins to select the PS configuration scheme. Any other Altera device that supports PS configuration can also be part of the chain as a configuration slave.

When connecting a serial configuration device to the Cyclone III device family in the multi-device AS configuration, you must connect a 25-Ω series resistor at the near end of the serial configuration device for DATA[0].

In the multi-device AS configuration, the board trace length between the serial configuration device to the master device of the Cyclone III device family must follow the recommendations in Table 9–9 on page 9–20. You must also connect the repeater buffers between the master and slave devices of the Cyclone III device family for DATA[0] and DCLK. All I/O inputs must maintain a maximum AC voltage of 4.1 V. The output resistance of the repeater buffers must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 9–7.

As shown in Figure 9–4 on page 9–15, the nSTATUS and CONF_DONE pins on all target devices are connected together with external pull-up resistors. These pins are open-drain bidirectional pins on the devices. When the first device asserts nCEO (after receiving all its configuration data), it releases its CONF_DONE pin. However, the subsequent devices in the chain keep this shared CONF_DONE line low until they receive their configuration data. When all target devices in the chain receive their configuration data and release CONF_DONE, the pull-up resistor drives a high level on this line and all devices simultaneously enter initialization mode.

Although you can cascade Cyclone III device family, serial configuration devices cannot be cascaded or chained together.

If the configuration bitstream size exceeds the capacity of a serial configuration device, you must select a larger configuration device, enable the compression feature, or both. When configuring multiple devices, the size of the bitstream is the sum of the individual devices configuration bitstreams.

Configuring Multiple Cyclone III Device Family with the Same Design

Certain designs require you to configure multiple Cyclone III device family with the same design through a configuration bitstream or a **.sof**. You can do this using the following methods:

- Multiple SRAM Object Files
- Single SRAM Object File

For both methods, the serial configuration devices cannot be cascaded or chained together.

Multiple SRAM Object Files

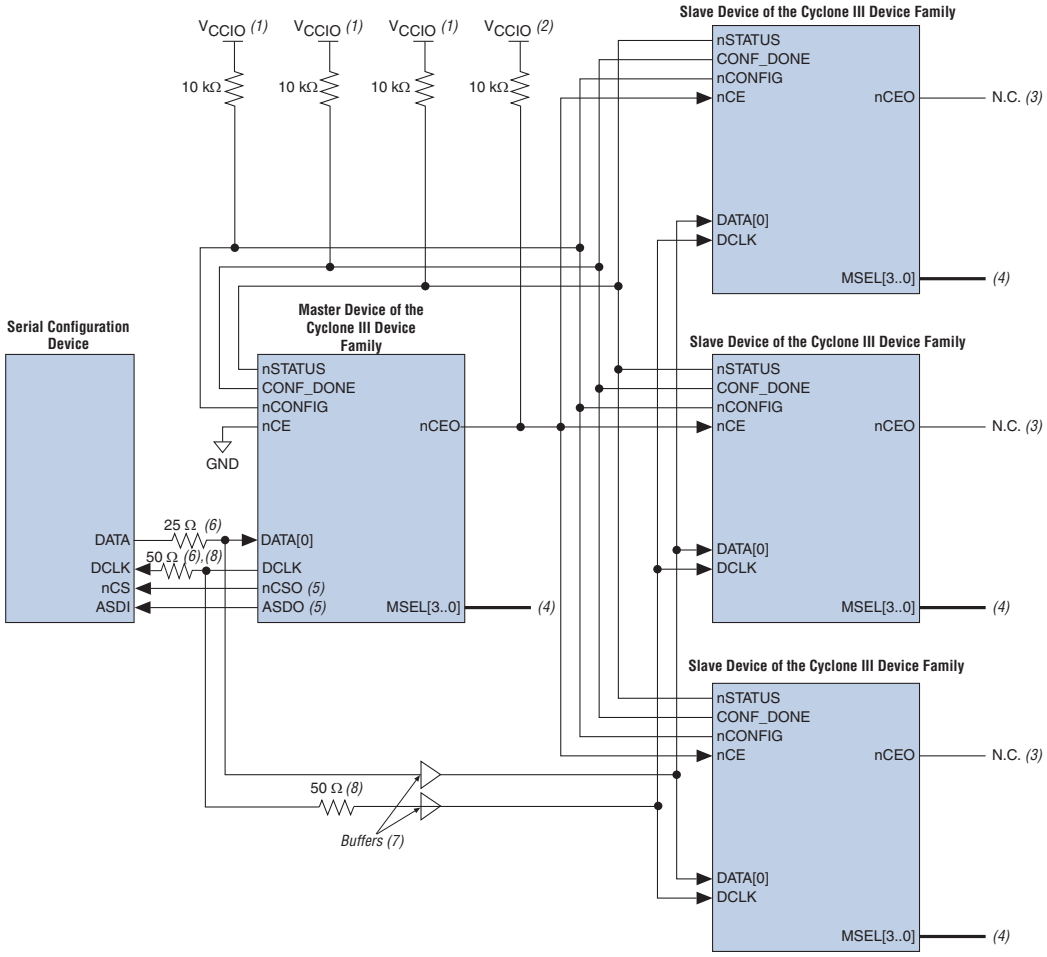
Two copies of the **.sof** are stored in the serial configuration device. Use the first copy to configure the master device of the Cyclone III device family and the second copy to configure all the remaining slave devices concurrently. All slave devices must be of the same density and package. The setup is similar to Figure 9–4 on page 9–15, in which the master device is set up in AS mode and the slave devices are set up in PS mode.

To configure four identical Cyclone III device family with the same **.sof**, you must set up the chain similar to Figure 9–5. The first device is the master device and its MSEL pins must be set to select the AS configuration. The other three slave devices are set up for concurrent configuration and their MSEL pins must be set to select the PS configuration. The nCEO pin from the master device drives the nCE input pins on all three slave devices, as well as the DATA and DCLK pins that connect in parallel to all four devices. During the first configuration cycle, the master device reads its configuration data from the serial configuration device while holding nCEO high. After completing its configuration cycle, the master device drives nCE low and sends the second copy of the configuration data to all three slave devices, configuring them simultaneously.

The advantage of using the setup in Figure 9–5 is that you can have a different **.sof** for the master device. However, all the slave devices must be configured with the same **.sof**. In this configuration method, you can either compress or uncompress the **.sof**s.

You can still use this method if the master and slave devices use the same **.sof**.

Figure 9–5. Multi-Device AS Configuration where the Devices Receive the Same Data with Multiple SRAM Object Files



Notes to Figure 9–5:

(1) Connect the pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) Connect the pull-up resistor to the V_{CCIO} supply voltage of the I/O bank in which the nCE pin resides.

(3) The nCEO pin is left unconnected or used as a user I/O pin when it does not feed the nCE pin of another device.

(4) The MSEL pin settings vary for different configuration voltage standards and POR time. You must set the master device in AS mode and the slave devices in PS mode. To connect MSEL[3..0] for the master device in AS mode and the slave devices in PS mode, refer to Table 9–7 on page 9–11. Connect the MSEL pins directly to V_{CCA} or GND.

(5) These are dual-purpose I/O pins. The nCSO pin functions as the FLASH_NCE pin in AP mode. The ASDO pin functions as the DATA[1] pin in other AP and FPP modes.

(6) Connect the series resistor at the near end of the serial configuration device.

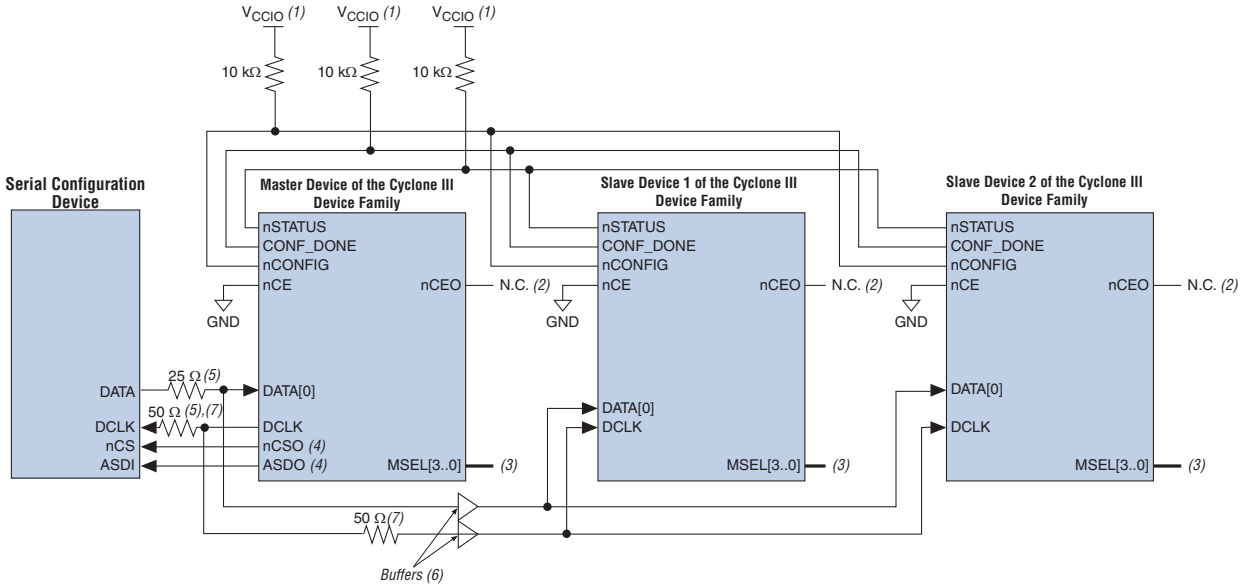
(7) Connect the repeater buffers between the master and slave devices for DATA[0] and DCLK. All I/O inputs must maintain a maximum AC voltage of 4.1 V. The output resistance of the repeater buffers must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 9–7.

(8) The 50-Ω series resistors are optional if the 3.3-V configuration voltage standard is applied. For optimal signal integrity, connect these 50-Ω series resistors if the 2.5- or 3.0-V configuration voltage standard is applied.

Single SRAM Object File

The second method configures both the master device and slave devices with the same **.sof**. The serial configuration device stores one copy of the **.sof**. This setup is shown in Figure 9–6 where the master is set up in AS mode and the slave devices are set up in PS mode. You must set up one or more slave devices in the chain. All the slave devices must be set up as shown in Figure 9–6.

Figure 9–6. Multi-Device AS Configuration where the Devices Receive the Same Data with a Single .sof



Notes to Figure 9–6:

(1) Connect the pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) The nCEO pin is left unconnected or used as a user I/O pin when it does not feed the nCE pin of another device.

(3) The MSEL pin settings vary for different configuration voltage standards and POR time. You must set the master device of the Cyclone III device family in AS mode and the slave devices in PS mode. To connect MSEL[3..0] for the master device in AS mode and slave devices in PS mode, refer to Table 9–7 on page 9–11. Connect the MSEL pins directly to V_{CCA} or GND.

(4) These are dual-purpose I/O pins. The nCSO pin functions as the FLASH_NCE pin in AP mode. The ASDO pin functions as the DATA[1] pin in other AP and FPP modes.

(5) Connect the series resistor at the near end of the serial configuration device.

(6) Connect the repeater buffers between the master and slave devices for DATA[0] and DCLK. All I/O inputs must maintain a maximum AC voltage of 4.1 V. The output resistance of the repeater buffers must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 9–7.

(7) The 50-Ω series resistors are optional if the 3.3-V configuration voltage standard is applied. For optimal signal integrity, connect these 50-Ω series resistors if the 2.5- or 3.0-V configuration voltage standard is applied.

In this setup, all the Cyclone III device family in the chain are connected for concurrent configuration. This can reduce the AS configuration time because all the Cyclone III device family is configured in one configuration cycle. Connect the nCE input pins of all the Cyclone III device family to ground. You can either leave the nCEO output pins on all the Cyclone III device family unconnected or use the nCEO output pins as normal user I/O pins. The DATA and DCLK pins are connected in parallel to all the Cyclone III device family.

Altera recommends putting a buffer before the DATA and DCLK output from the master device to avoid signal strength and integrity issues. The buffer must not significantly change the DATA-to-DCLK relationships or delay them with respect to other AS signals (ASDI and nCS). Also, the buffer must only drive the slave devices to ensure that the timing between the master device and the serial configuration device is unaffected.

This configuration method supports both compressed and uncompressed **.sof**s. Therefore, if the configuration bitstream size exceeds the capacity of a serial configuration device, you can enable the compression feature in the **.sof** or you can select a larger serial configuration device.

Guidelines for Connecting Serial Configuration Device to Cyclone III Device Family on AS Interface

For single- and multi-device AS configurations, the board trace length and loading between the supported serial configuration device and Cyclone III device family must follow the recommendations listed in Table 9–9.

Table 9–9. Maximum Trace Length and Loading for the AS Configuration

Cyclone III Device Family AS Pins	Maximum Board Trace Length from the Cyclone III Device Family to the Serial Configuration Device (Inches)	Maximum Board Load (pF)
DCLK	10	15
DATA[0]	10	30
nCSO	10	30
ASDO	10	30

Estimating AS Configuration Time

AS configuration time is dominated by the time it takes to transfer data from the serial configuration device to the Cyclone III device family. This serial interface is clocked by the Cyclone III device family DCLK output (generated from an internal oscillator). Equation 9–2 and Equation 9–3 show the configuration time estimation for the Cyclone III device family.

Equation 9–2.

$$\text{RBF Size} \times \left(\frac{\text{maximum DCLK period}}{\text{1 bit}}\right) = \text{estimated maximum configuration time}$$

Equation 9–3.

$$3{,}500{,}000 \text{ bits} \times \left(\frac{50 \text{ ns}}{\text{1 bit}}\right) = 175 \text{ ms}$$

To estimate the typical configuration time, use the typical DCLK period shown in Figure 9–7 on page 9–22. With a typical DCLK period of 33.33 ns, the typical configuration time is 116.7 ms. Enabling compression reduces the amount of configuration data that is sent to the Cyclone III device family, which also reduces configuration time. On average, compression reduces configuration time by 50%.

Programming Serial Configuration Devices

Serial configuration devices are non-volatile, flash memory-based devices. You can program these devices in-system using the USB-Blaster™ or ByteBlaster™ II download cable. Alternatively, you can program them using the Altera Programming Unit (APU), supported third-party programmers, or a microprocessor with the SRunner software driver.

You can perform in-system programming of serial configuration devices using the AS programming interface. During in-system programming, the download cable disables device access to the AS interface by driving the nCE pin high. Cyclone III device family is also held in reset by a low level on nCONFIG. After programming is complete, the download cable releases nCE and nCONFIG, allowing the pull-down and pull-up resistors to drive GND and V_{CC}, respectively.

To perform in-system programming of a serial configuration device using the AS programming interface, the diodes and capacitors must be placed as close as possible to the Cyclone III device family. Ensure that the diodes and capacitors maintain a maximum AC voltage of 4.1 V (Figure 9–7).

If you wish to use the same setup shown in Figure 9–7 to perform in-system programming of a serial configuration device and single- or multi-device AS configuration, you do not need a series resistor on the DATA line at the near end of the serial configuration device. The existing diodes and capacitors are sufficient.

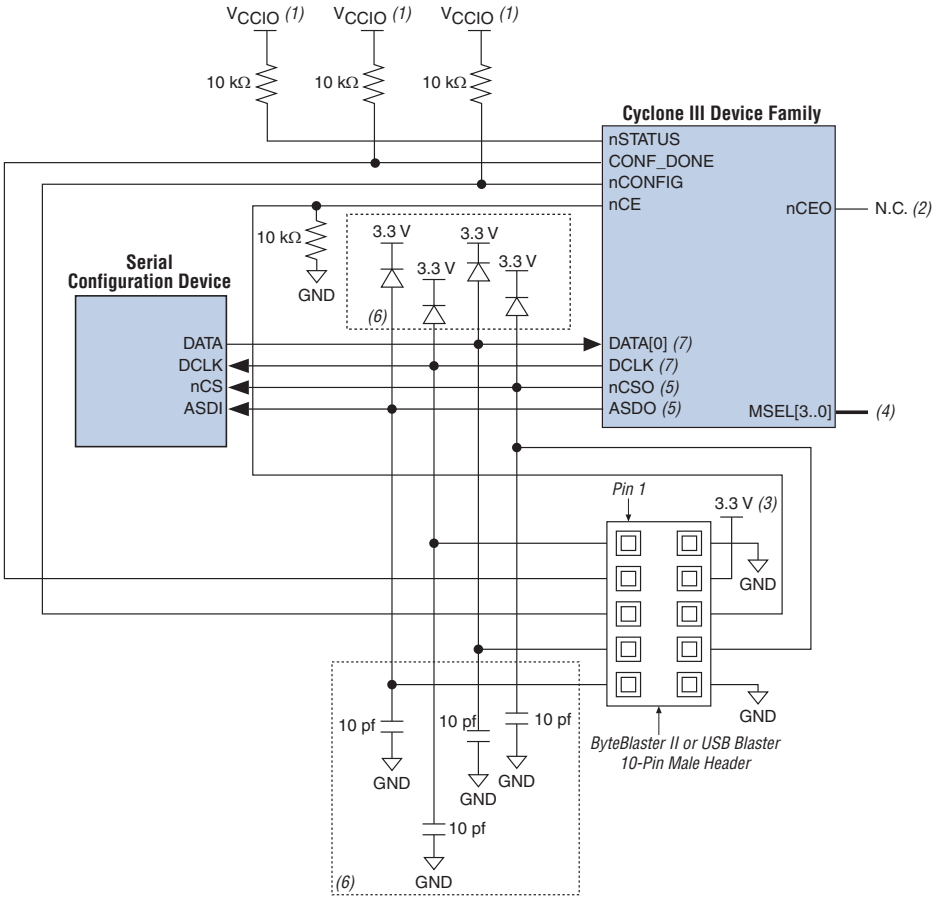
Altera has developed the Serial FlashLoader (SFL), a JTAG-based in-system programming solution for Altera serial configuration devices. The SFL is a bridge design for the Cyclone III device family that uses its JTAG interface to access the EPCS JIC (JTAG Indirect Configuration Device Programming) file and then uses the AS interface to program the EPCS device. Both the JTAG interface and AS interface are bridged together inside the SFL design.

For more information about implementing the SFL with Cyclone III device family, refer to *AN 370: Using the Serial FlashLoader with the Quartus II Software*.

For more information about the USB-Blaster download cable, refer to the *USB-Blaster Download Cable User Guide*. For more information about the ByteBlaster II download cable, refer to the *ByteBlaster II Download Cable User Guide*.

Figure 9–7 shows the download cable connections to the serial configuration device.

Figure 9–7. In-System Programming of Serial Configuration Devices



Notes to Figure 9–7:

(1) Connect these pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) The nCEO pin is left unconnected or used as a user I/O pin when it does not feed the nCE pin of another device.

(3) Power up the V_{CC} of the ByteBlaster II or USB-Blaster download cable with the 3.3-V supply.

(4) The MSEL pin settings vary for different configuration voltage standards and POR time. To connect MSEL[3..0], refer to Table 9–7 on page 9–11. Connect the MSEL pins directly to V_{CCA} or ground.

(5) These are dual-purpose I/O pins. This nCSO pin functions as the FLASH_NCE pin in AP mode. The ASDO pin functions as the DATA[1] pin in other AP and FPP modes.

(6) The diodes and capacitors must be placed as close as possible to the Cyclone III device family. Ensure that the diodes and capacitors maintain a maximum AC voltage of 4.1 V. The external diodes and capacitors are required to prevent damage to the Cyclone III device family AS configuration input pins due to possible overshoot when programming the serial configuration device using a download cable. For effective voltage clamping, Altera recommends using the Schottky diode, which has a relatively lower forward diode voltage (VF) than the switching and Zener diodes. For more information about the interface guidelines using Schottky diodes, refer to *AN 523: Cyclone III Configuration Interface Guidelines with EPCS Devices.*

(7) When cascading Cyclone III device family in a multi-device AS configuration, connect the repeater buffers between the master and slave devices for DATA[0] and DCLK. All I/O inputs must maintain a maximum AC voltage of 4.1 V. The output resistance of the repeater buffers must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 9–7.

You can use the Quartus II software with the APU and the appropriate configuration device programming adapter to program serial configuration devices. All serial configuration devices are offered in an 8- or 16-pin small outline integrated circuit (SOIC) package.

In production environments, serial configuration devices are programmed using multiple methods. Altera programming hardware or other third-party programming hardware is used to program blank serial configuration devices before they are mounted onto PCBs. Alternatively, you can use an on-board microprocessor to program the serial configuration device in-system by porting the reference C-based SRunner software driver provided by Altera.

A serial configuration device is programmed in-system by an external microprocessor using the SRunner software driver. The SRunner software driver is a software driver developed for embedded serial configuration device programming, which is easily customized to fit in different embedded systems. The SRunner software driver is able to read a Raw Programming Data (**.rpd**) file and write to the serial configuration devices. The serial configuration device programming time using the SRunner software driver is comparable to the programming time with the Quartus II software.

For more information about the SRunner software driver, refer to *AN 418: SRunner: An Embedded Solution for Serial Configuration Device Programming* and the source code at the Altera website (www.altera.com).

AP Configuration (Supported Flash Memories)

The AP configuration scheme is for Cyclone III devices only. In the AP configuration scheme, Cyclone III devices are configured using commodity 16-bit parallel flash memory. These external non-volatile configuration devices are industry standard microprocessor flash memories. The flash memories provide a fast interface to access the configuration data. The speed-up in configuration time is mainly due to the 16-bit wide parallel data bus, which is used to retrieve data from the flash memory.

Some of the smaller Cyclone III devices or package options do not support the AP configuration scheme and do not have the `MSEL[3]` pin. Table 9–10 lists the supported AP configuration scheme for each Cyclone III device.

Table 9–10. Supported AP Configuration Scheme for Cyclone III Devices

Device	Package Options								
	E144	M164	Q240	F256	F324	F484	F780	U256	U484
EP3C5	—	—	—	—	—	—	—	—	—
EP3C10	—	—	—	—	—	—	—	—	—
EP3C16	—	—	—	—	—	✓	—	—	✓
EP3C25	—	—	—	—	✓	—	—	—	—
EP3C40	—	—	—	—	✓	✓	✓	—	✓
EP3C55	—	—	—	—	—	✓	✓	—	✓
EP3C80	—	—	—	—	—	✓	✓	—	✓
EP3C120	—	—	—	—	—	✓	✓	—	—

During device configuration, Cyclone III devices read configuration data using the parallel interface and configure their SRAM cells. This scheme is referred to as the AP configuration scheme because the device controls the configuration interface. This scheme contrasts with the FPP configuration scheme, where an external host controls the interface.

AP Configuration Supported Flash Memory

The AP configuration controller in Cyclone III devices is designed to interface with the Numonyx StrataFlash® Embedded Memory P30 flash family and the Numonyx StrataFlash Embedded Memory P33 flash family, which are two industry standard flash families. Unlike serial configuration devices, both of the flash families supported in the AP configuration scheme are designed to interface with microprocessors. By configuring from an industry standard microprocessor flash which allows access to the flash after entering user mode, the AP configuration scheme allows you to combine configuration data and user data (microprocessor boot code) on the same flash memory.

The Numonyx P30 and P33 flash families support a continuous synchronous burst read mode at 40 MHz DCLK frequency for reading data from the flash. Additionally, the Numonyx P30 and P33 flash families have identical pin-out and adopt similar protocols for data access.

Cyclone III devices use a 40-MHz oscillator for the AP configuration scheme.

Table 9–11 lists the supported families of the commodity parallel flash for the AP configuration scheme.

Table 9–11. Supported Commodity Flash for the AP Configuration Scheme for Cyclone III Devices *(Note 1)*

Flash Memory Density	Numonyx P30 Flash Family *(2)*	Numonyx P33 Flash Family *(3)*
64 Mbit	✓	✓
128 Mbit	✓	✓
256 Mbit	✓	✓

Notes to Table 9–11:

(1) The AP configuration scheme only supports flash memory speed grades of 40 MHz and above.
(2) 3.3- , 3.0-, 2.5-, and 1.8-V I/O options are supported for the Numonyx P30 flash family.
(3) 3.3-, 3.0- and 2.5-V I/O options are supported for the Numonyx P33 flash family.

The AP configuration scheme of Cyclone III devices supports the Numonyx P30 and P33 family 64-, 128-, and 256-Mbit flash memories. Configuring Cyclone III devices from the Numonyx P30 and P33 family 512-Mbit flash memory is possible, but you must properly drive the extra address and nCS pins as required by these flash memories.

You must refer to the respective flash data sheets to check for supported speed grades and package options.

The AP configuration scheme in Cyclone III devices supports flash speed grades of 40 MHz and above. However, the AP configuration for all these speed grades must be capped at 40 MHz. The advantage of faster speed grades is realized when your design in the Cyclone III device accesses flash memory in user mode.

For more information about the operation of the Numonyx StrataFlash Embedded Memory P30 and P33 flash memories, search for the keyword "P30" or "P33" on the Numonyx website (www.numonyx.com) to obtain the P30 or P33 family data sheet.

Single-Device AP Configuration

The following groups of interface pins are supported in Numonyx P30 and P33 flash memories:

- Control pins
- Address pins
- Data pins

Following are the control signals from the supported parallel flash memories:

- CLK
- active-low reset (RST#)
- active-low chip enable (CE#)
- active-low output enable (OE#)
- active-low address valid (ADV#)
- active-low write enable (WE#)

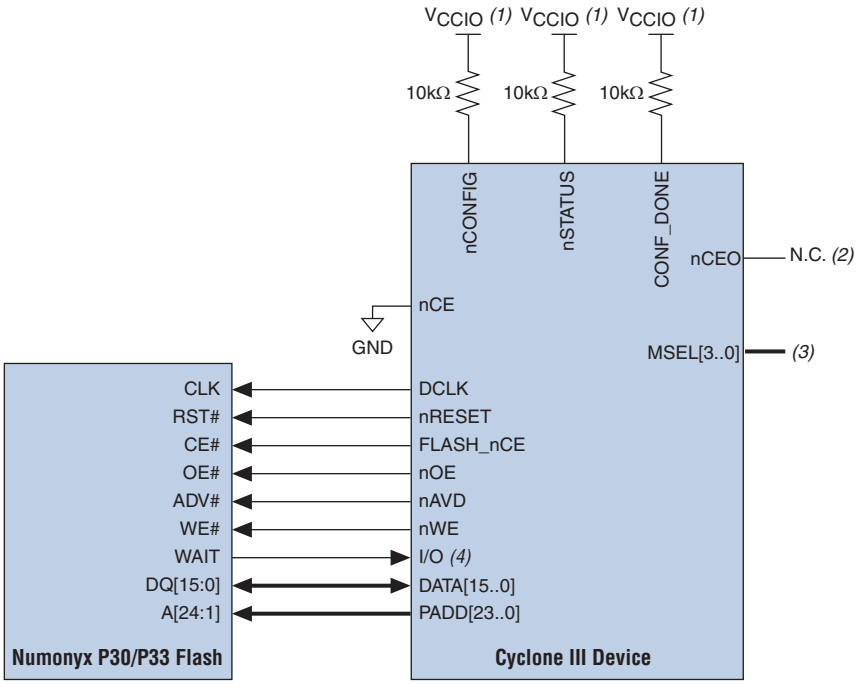
The supported parallel flash memories output a control signal (WAIT) to Cyclone III devices to indicate when synchronous data is ready on the data bus. Cyclone III devices have a 24-bit address bus connecting to the address bus (A[24:1]) of the flash memory. A 16-bit bidirectional data bus (DATA[15..0]) provides data transfer between the Cyclone III device and the flash memory.

The following are the control signals from the Cyclone III device to flash memory:

- DCLK
- nRESET
- FLASH_nCE
- nOE
- nAVD
- nWE

The interface for the Numonyx P30 flash memory and P33 flash memory connects to Cyclone III device pins, as shown in Figure 9–8.

Figure 9–8. Single-Device AP Configuration Using Numonyx P30 and P33 Flash Memory



Notes to Figure 9–8:

(1) Connect the pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) The `nCEO` pin is left unconnected or used as a user I/O pin when it does not feed the `nCE` pin of another device.

(3) The MSEL pin settings vary for different configuration voltage standards and POR time. To connect `MSEL[3..0]`, refer to Table 9–7 on page 9–11. Connect the MSEL pins directly to V_{CCA} or GND.

(4) The AP configuration ignores the `WAIT` signal during configuration mode. However, if you are accessing flash during user mode with user logic, you can optionally use a normal I/O to monitor the `WAIT` signal from the Numonyx P30 or P33 flash.

In a single-device AP configuration, the maximum board loading and board trace length between the supported parallel flash and Cyclone III devices must follow the recommendations listed in Table 9–12 on page 9–30.

If you use the AP configuration scheme for Cyclone III devices, the V_{CCIO} of I/O banks 1, 6, 7, and 8 must be 3.3, 3.0, 2.5, or 1.8 V. Altera does not recommend using the level shifter between the Numonyx P30/P33 flash and the Cyclone III device in the AP configuration scheme.

There are no series resistors required in the AP configuration mode for Cyclone III devices when using the Numonyx flash at 2.5-, 3.0-, and 3.3-V I/O standard. The output buffer of the Numonyx P30 IBIS model does not overshoot above 4.1 V. Thus, series resistors are not required for the 2.5-, 3.0-, and 3.3-V AP configuration option. However, if there are any other devices sharing the same flash I/Os with Cyclone III devices, all shared pins are still subject to the 4.1-V limit and may require series resistors.

The default read mode of the supported parallel flash memory and all writes to the parallel flash memory are asynchronous. Both the parallel flash families support a synchronous read mode, with data supplied on the positive edge of DCLK.

- nRESET is an active-low hard reset
- FLASH_nCE is an active-low chip enable
- nOE is an active-low output enable for the DATA[15..0] bus and WAIT pin
- nAVD is an active-low address valid signal and is used to write addresses into the flash
- nWE is an active-low write enable and is used to write data into the flash
- PADD[23..0] bus is the address bus supplied to the flash
- DATA[15..0] bus is a bidirectional bus used to supply and read data to and from the flash, with the flash output controlled by nOE

The serial clock (DCLK) generated by Cyclone III devices controls the entire configuration cycle and provides timing for the parallel interface. Cyclone III devices use a 40-Mhz internal oscillator to generate DCLK. The oscillator is the same oscillator used in the AS configuration scheme. The active DCLK output frequency is listed in Table 9–8 on page 9–13.

Multi-Device AP Configuration

You can cascade multiple Cyclone III devices using the chip-enable (nCE) and chip-enable-out (nCEO) pins. The first device in the chain must have its nCE pin connected to GND. Connect its nCEO pin to the nCE pin of the next device in the chain. Use an external 10-kΩ pull-up resistor to pull the nCEO signal high to its V_{CCIO} level to help the internal weak pull-up resistor. When the first device captures all its configuration data from the bitstream, it drives the nCEO pin low, enabling the next device in the chain. You can leave the nCEO pin of the last device unconnected or use it as a user I/O pin after configuration if the last device in the chain is a Cyclone III device. The nCONFIG, nSTATUS, CONF_DONE, DCLK, DATA[15..8], and DATA[7..0] pins of each device in the chain are connected (Figure 9–9 on page 9–28 and Figure 9–10 on page 9–29).

The first Cyclone III device in the chain, as shown in Figure 9–9 on page 9–28 and Figure 9–10 on page 9–29, is the configuration master device and controls the configuration of the entire chain. Connect its MSEL pins to select the AP configuration scheme. The remaining Cyclone III devices are used as configuration slaves. Connect their MSEL pins to select the FPP configuration scheme. Any other Altera device that supports FPP configuration can also be part of the chain as a configuration slave.

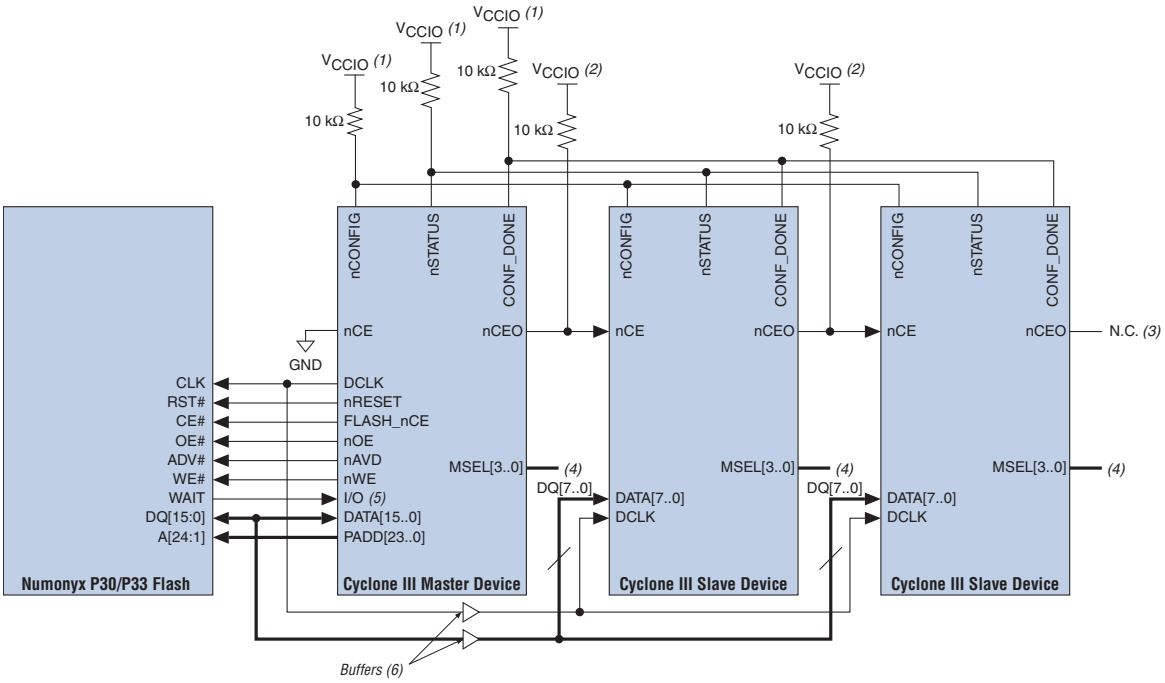
The following are the configurations for the DATA[15..0] bus in a multi-device AP configuration:

- Byte-wide multi-device AP configuration
- Word-wide multi-device AP configuration

Byte-Wide Multi-Device AP Configuration

The simpler method for multi-device AP configuration is the byte-wide multi-device AP configuration. In the byte-wide multi-device AP configuration, the LSB of the DATA[7..0] pin from the flash and master device (set to the AP configuration scheme) is connected to the slave devices set to the FPP configuration scheme, as shown in Figure 9–9.

Figure 9–9. Byte-Wide Multi-Device AP Configuration



Notes to Figure 9–9:

(1) Connect the pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) Connect the pull-up resistor to the V_{CCIO} supply voltage of the I/O bank in which the nCE pin resides.

(3) The nCEO pin is left unconnected or used as a user I/O pin when it does not feed the nCE pin of another device.

(4) The MSEL pin settings vary for different configuration voltage standards and POR time. You must set the master device in AP mode and the slave devices in FPP mode. To connect MSEL[3..0] for the master device in AP mode and the slave devices in FPP mode, refer to Table 9–7 on page 9–11. Connect the MSEL pins directly to V_{CCA} or GND.

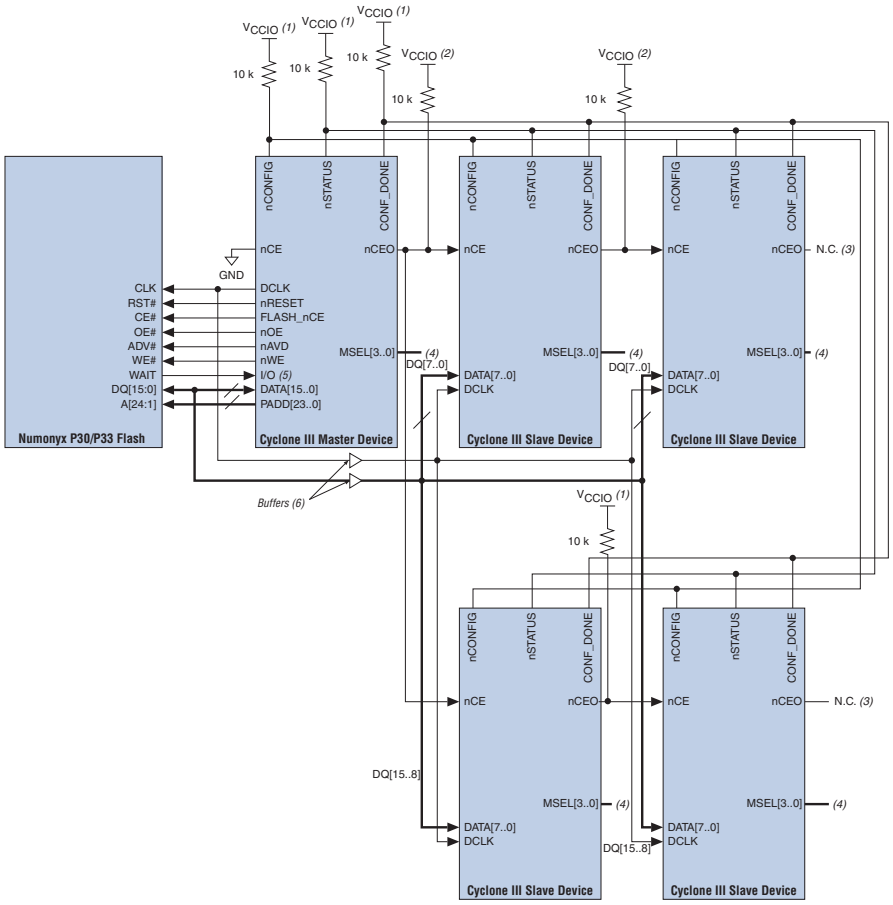
(5) The AP configuration ignores the WAIT signal during configuration mode. However, if you are accessing flash during user mode with user logic, you can optionally use the normal I/O to monitor the WAIT signal from the Numonyx P30 or P33 flash.

(6) Connect the repeater buffers between the master device and slave devices for DATA[15..0] and DCLK. All I/O inputs must maintain a maximum AC voltage of 4.1 V. The output resistance of the repeater buffers must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 9–7.

Word-Wide Multi-Device AP Configuration

The more efficient setup is one in which some of the slave devices are connected to the LSB of DATA[7..0] and the remaining slave devices are connected to the MSB of DATA[15..8]. In the word-wide multi-device AP configuration, the nCEO pin of the master device enables two separate daisy-chains of slave devices, allowing both chains to be programmed concurrently, as shown in Figure 9–10.

Figure 9–10. Word-Wide Multi-Device AP Configuration



Notes to Figure 9–10:

(1) Connect the pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) Connect the pull-up resistor to the V_{CCIO} supply voltage of the I/O bank in which the nCE pin resides.

(3) The nCEO pin is left unconnected or used as a user I/O pin when it does not feed the nCE pin of another device.

(4) The MSEL pin settings vary for different configuration voltage standards and POR time. You must set the master device in AP mode and the slave devices in FPP mode. To connect MSEL[3..0] for the master device in AP mode and the slave devices in FPP mode, refer to Table 9–7 on page 9–11. Connect the MSEL pins directly to V_{CCA} or GND.

(5) The AP configuration ignores the WAIT signal during configuration mode. However, if you are accessing flash during user mode with user logic, you can optionally use the normal I/O pin to monitor the WAIT signal from the Numonyx P30 or P33 flash.

(6) Connect the repeater buffers between the Cyclone III master device and slave devices for DATA[15..0] and DCLK. All I/O inputs must maintain a maximum AC voltage of 4.1 V. The output resistance of the repeater buffers must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 9–7.

In a multi-device AP configuration, the board trace length between the parallel flash and the master device must follow the recommendations listed in Table 9–12.

As shown in Figure 9–9 and Figure 9–10, the nSTATUS and CONF_DONE pins on all target devices are connected together with external pull-up resistors. These pins are open-drain bidirectional pins on the devices. When the first device asserts nCEO (after receiving all its configuration data), it releases its CONF_DONE pin. However, the subsequent devices in the chain keep this shared CONF_DONE line low until they receive their configuration data. When all target devices in the chain receive their configuration data and release CONF_DONE, the pull-up resistor drives a high level on this line and all devices simultaneously enter initialization mode.

Guidelines for Connecting Parallel Flash to Cyclone III Devices for the AP Interface

For the single- and multi-device AP configuration, the board trace length and loading between the supported parallel flash and Cyclone III devices must follow the recommendations listed in Table 9–12. These recommendations also apply to an AP configuration with multiple bus masters.

Table 9–12. Maximum Trace Length and Loading for the AP Configuration

Cyclone III AP Pins	Maximum Board Trace Length from the Cyclone III Device to the Flash Device (Inches)	Maximum Board Load (pF)
DCLK	6	15
DATA[15..0]	6	30
PADD[23..0]	6	30
nRESET	6	30
Flash_nCE	6	30
nOE	6	30
nAVD	6	30
nWE	6	30
I/O *(1)*	6	30

Note to Table 9–12:

(1) The AP configuration ignores the WAIT signal from the flash during configuration mode. However, if you are accessing flash during user mode with user logic, you can optionally use the normal I/O to monitor the WAIT signal from the Numonyx P30 or P33 flash.

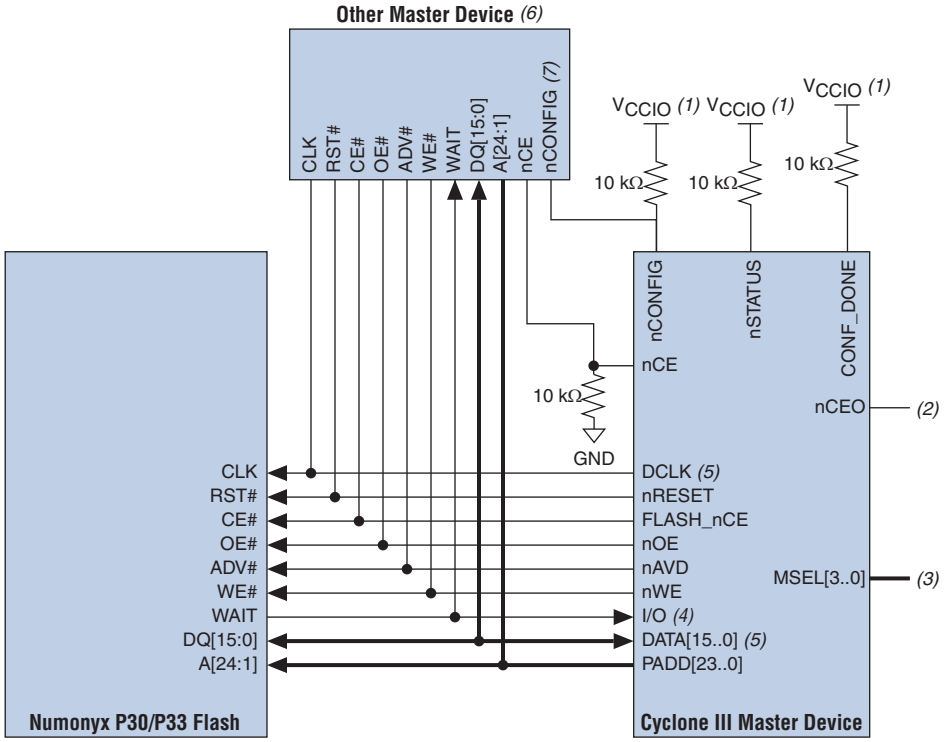
Configuring With Multiple Bus Masters

Similar to the AS configuration scheme, the AP configuration scheme supports multiple bus masters for the parallel flash. For another master to take control of the AP configuration bus, the master must assert nCONFIG low for at least 500 ns to reset the master Cyclone III device and override the weak 10 kΩ pull-down resistor on the nCE pin. This resets the master Cyclone III device and causes it to tri-state its AP configuration bus. The other master then takes control of the AP configuration bus. After the other master is done, it releases the AP configuration bus, then releases the nCE pin, and finally pulses nCONFIG low to restart the configuration.

In the AP configuration scheme, multiple masters share the parallel flash. Similar to the AS configuration scheme, the bus control is negotiated by the nCE pin.

Figure 9–11 shows the AP configuration with multiple bus masters.

Figure 9–11. AP Configuration with Multiple Bus Masters



Notes to Figure 9–11:

(1) Connect the pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) The `nCEO` pin is left unconnected or used as a user I/O pin when it does not feed the `nCE` pin of another device.

(3) The MSEL pin settings vary for different configuration voltage standards and POR time. To connect `MSEL[3..0]`, refer to Table 9–7 on page 9–11. Connect the MSEL pins directly to V_{CCA} or GND.

(4) The AP configuration ignores the `WAIT` signal during configuration mode. However, if you are accessing flash during user mode with user logic, you can optionally use the normal I/O to monitor the `WAIT` signal from the Numonyx P30 or P33 flash.

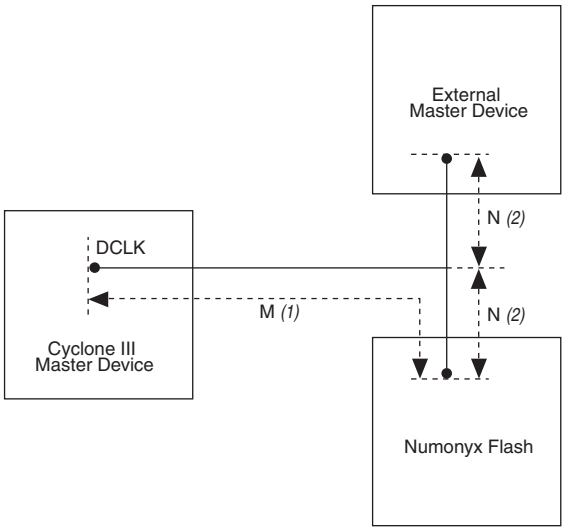
(5) When cascading Cyclone III devices in a multi-device AP configuration, connect the repeater buffers between the master device and slave devices for `DATA[15..0]` and `DCLK`. All I/O inputs must maintain a maximum AC voltage of 4.1 V. The output resistance of the repeater buffers must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 9–7.

(6) The other master device must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 9–7.

(7) The other master device can pulse `nCONFIG` if it is under system control rather than tied to V_{CCIO}.

Figure 9–12 shows the recommended balanced star routing for multiple bus master interfaces to minimize signal integrity issue.

Figure 9–12. Balanced Star Routing



Notes to Figure 9–12:

(1) Altera does not recommend *M* to exceed six inches as listed in Table 9–12 on page 9–30.

(2) Altera recommends using a balanced star routing. Try to keep the *N* length equal and as short as possible to minimize reflection noise from the transmission line. The *M* length is applicable for this setup.

Estimating the AP Configuration Time

AP configuration time is dominated by the time it takes to transfer data from the parallel flash to the Cyclone III devices. This parallel interface is clocked by the Cyclone III DCLK output (generated from an internal oscillator). As listed in Table 9–8 on page 9–13, the DCLK minimum frequency when using the 40-MHz oscillator is 20 MHz (50 ns). In word-wide cascade programming, the DATA[15..0] bus transfers a 16-bit word and essentially cuts configuration time to approximately 1/16 of the AS configuration time. Therefore, the maximum configuration time estimation for an EP3C40 device (9,600,000 bits of uncompressed data) is defined in Equation 9–4 and Equation 9–5.

Equation 9–4.

$$\text{RBF Size} \times \left(\frac{\text{maximum DCLK period}}{\text{16 bits per DCLK cycle}} \right) = \text{estimated maximum configuration time}$$

Equation 9–5.

$$9{,}600{,}000 \text{ bits} \times \left(\frac{50 \text{ ns}}{16 \text{ bits}} \right) = 30 \text{ ms}$$

To estimate a typical configuration time, use the typical DCLK period listed in Table 9–8 on page 9–13. With a typical DCLK period of 33.33 ns, the typical configuration time is 20 ms.

Programming Parallel Flash Memories

Supported parallel flash memories are external non-volatile configuration devices. They are industry standard microprocessor flash memories. For more information about the supported families for the commodity parallel flash, refer to Table 9–11 on page 9–24.

Cyclone III devices in a single- or multiple-device chains support in-system parallel flash programming with the JTAG interface using the flash loader megafunction. For Cyclone III devices, the board-intelligent host or download cable uses four JTAG pins to program the parallel flash in system, even if the host or download cable cannot access the configuration pins of the parallel flash.

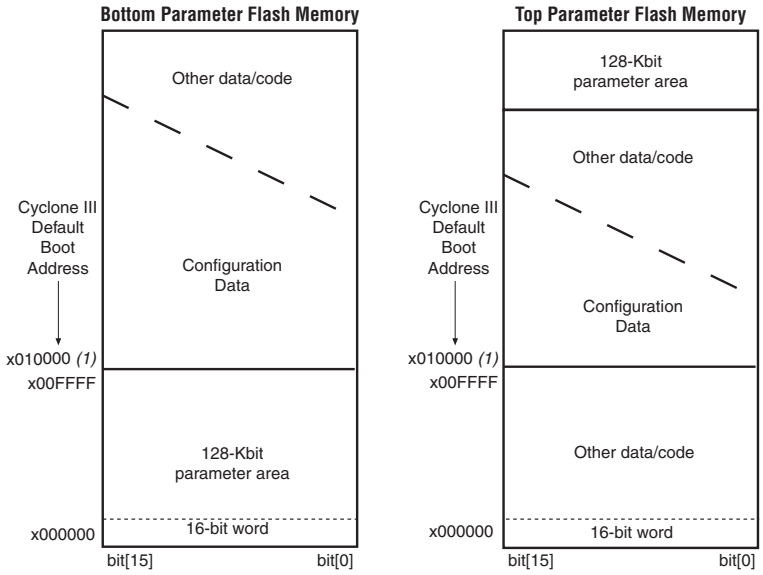
For more information about using the JTAG pins on Cyclone III devices to program the parallel flash in-system, refer to *AN 478: Using FPGA-Based Parallel Flash Loader (PFL) with the Quartus II Software*.

In the AP configuration scheme, the default configuration boot address is **0×010000** when represented in 16-bit word addressing in the supported parallel flash memory (Figure 9–13). In the Quartus II software, the default configuration boot address is 0x020000 because it is represented in 8-bit byte addressing. Cyclone III devices configure from word address 0x010000, which is equivalent to byte address 0x020000.

The Quartus II software uses byte addressing for the default configuration boot address. You must set the start address field to **0x020000**.

The default configuration boot addressing allows the system to use special parameter blocks in the flash memory map. Parameter blocks are at the top or bottom of the memory map. The configuration boot address in the AP configuration scheme is shown in Figure 9–13. You can change the default configuration default boot address **0x010000** to any desired address using the `APFC_BOOT_ADDR` JTAG instruction. For more information about the `APFC_BOOT_ADDR` JTAG instruction, refer to "JTAG Instructions" on page 9–61.

Figure 9–13. Configuration Boot Address in AP Flash Memory Map



Note to Figure 9–13:

(1) The default configuration boot address is x010000 when represented in 16-bit word addressing.

PS Configuration

You can perform PS configuration on Cyclone III device family with an external intelligent host, such as a MAX II device, microprocessor with flash memory, or a download cable. In the PS scheme, an external host controls the configuration. Configuration data is clocked into the target Cyclone III device family using the `DATA[0]` pin at each rising edge of `DCLK`.

If your system already contains a common flash interface (CFI) flash memory, you can use it for the Cyclone III device family configuration storage as well. The MAX II PFL feature provides an efficient method to program CFI flash memory devices through the JTAG interface and provides the logic to control the configuration from the flash memory device to the Cyclone III device family. Both PS and FPP configuration schemes are supported using the PFL feature.

For more information about the PFL, refer to *AN 386: Using the Parallel Flash Loader with the Quartus II Software*.

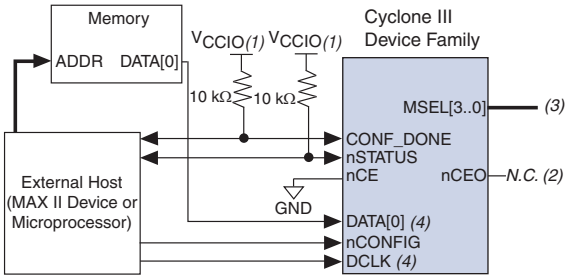
Cyclone III device family does not support enhanced configuration devices for PS or FPP configurations.

PS Configuration Using an External Host

In the PS configuration scheme, you can use an intelligent host such as MAX II or microprocessor that controls the transfer of configuration data from a storage device, such as flash memory, to the target Cyclone III device family. You can store the configuration data in **.rbf**, **.hex**, or **.ttf** format.

Figure 9–14 shows the configuration interface connections between a Cyclone III device family and an external host device for a single-device configuration.

Figure 9–14. Single-Device PS Configuration Using an External Host



Notes to Figure 9–14:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for the device. V_{CC} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

(2) The nCEO pin is left unconnected or used as a user I/O pin when it does not feed the nCE pin of another device.

(3) The MSEL pin settings vary for different configuration voltage standards and POR time. To connect MSEL[3..0], refer to Table 9–7 on page 9–11. Connect the MSEL pins directly to V_{CCA} or ground.

(4) All I/O inputs must maintain a maximum AC voltage of 4.1 V. DATA[0] and DCLK must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 9–7.

To begin configuration, the external host device must generate a low-to-high transition on the nCONFIG pin. When nSTATUS is pulled high, the external host device must place the configuration data one bit at a time on the DATA[0] pin. If you are using configuration data in a **.rbf**, **.ttf**, or **.hex** file, you must first send the LSB of each data byte. For example, if the **.rbf** contains the byte sequence 02 1B EE 01 FA, the serial bitstream you must send to the device is:

0100-0000 1101-1000 0111-0111 1000-0000 0101-1111

Cyclone III device family receives configuration data on the DATA[0] pin and the clock is received on the DCLK pin. Data is latched into the device on the rising edge of DCLK. Data is continuously clocked into the target device until CONF_DONE goes high and the device enters the initialization state.

Two DCLK falling edges are required after CONF_DONE goes high to begin device initialization.

The INIT_DONE pin is released and pulled high when initialization is complete. The external host device must be able to detect this low-to-high transition which signals the device has entered user mode. When initialization is complete, the device enters user mode. In user mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

To ensure DCLK and DATA[0] are not left floating at the end of configuration, the MAX II device must drive them either high or low, whichever is convenient on your board. The DATA[0] pin is available as a user I/O pin after configuration. In the PS scheme, the DATA[0] pin is tri-stated by default in user mode and must be driven by the external host device. To change this default option in the Quartus II software, select the **Dual-Purpose Pins** tab of the **Device and Pin Options** dialog box.

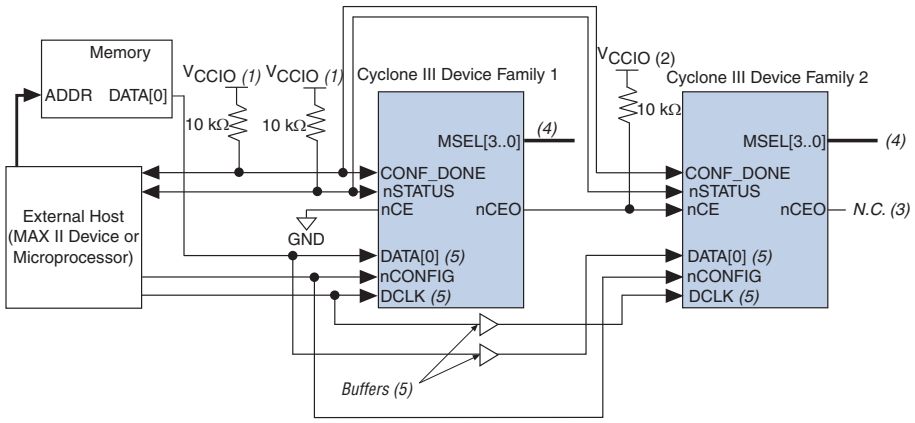
The configuration clock (DCLK) speed must be below the specified system frequency to ensure correct configuration (Figure 9–19 on page 9–42). No maximum DCLK period exists, which means you can pause configuration by halting DCLK for an indefinite amount of time.

If a configuration error occurs during configuration and the **Auto-restart configuration after error** option is turned on, the Cyclone III device family releases nSTATUS after a reset time-out period (a maximum of 230 µs). After nSTATUS is released and pulled high by a pull-up resistor, the external host device tries to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the external host device must generate a low-to-high transition (with a low pulse of at least 500 ns) on nCONFIG to restart the configuration process.

The external host device can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The CONF_DONE pin must be monitored by the external device to detect errors and to determine when the programming is complete. If all configuration data is sent, but CONF_DONE or INIT_DONE has not gone high, the external device must reconfigure the target device.

Figure 9–15 shows how to configure multiple devices using an external host device. This circuit is similar to the PS configuration circuit for a single device, except that the Cyclone III device family is cascaded for multi-device configuration.

Figure 9–15. Multi-Device PS Configuration Using an External Host



Notes to Figure 9–15:

(1) The pull-up resistor must be connected to a supply that provides an acceptable input signal for all devices in the chain. V_{CC} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

(2) Connect the pull-up resistor to the V_{CCIO} supply voltage of the I/O bank in which the nCE pin resides.

(3) The nCEO pin is left unconnected or used as a user I/O pin when it does not feed the nCE pin of another device.

(4) The MSEL pin settings vary for different configuration voltage standards and POR time. To connect MSEL[3..0], refer to Table 9–7 on page 9–11. Connect the MSEL pins directly to V_{CCA} or ground.

(5) All I/O inputs must maintain a maximum AC voltage of 4.1 V. DATA[0] and DCLK must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 9–7.

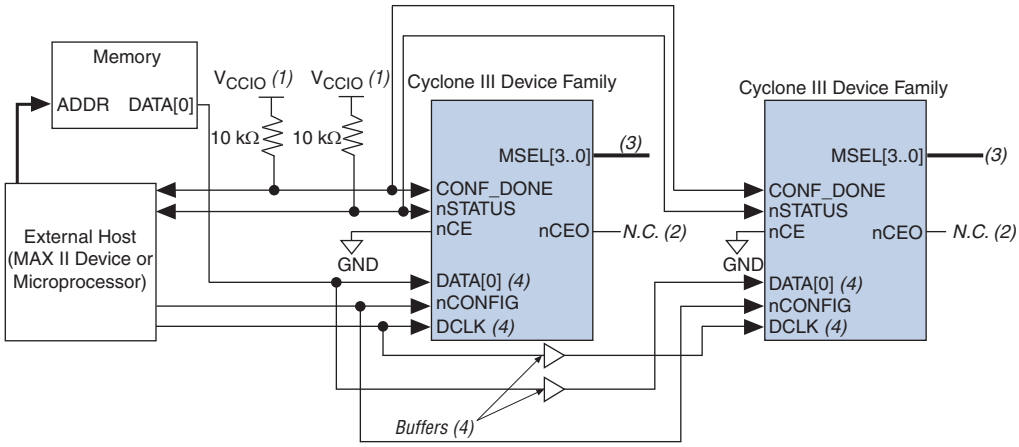
In a multi-device PS configuration, the nCE pin of the first device is connected to GND while its nCEO pin is connected to the nCE pin of the next device in the chain. The nCE input of the last device comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the nCE pin of the second device, which prompts the second device to begin configuration. The second device in the chain begins configuration in one clock cycle. Therefore, the transfer of data destinations is transparent to the external host device. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[0], and CONF_DONE) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

If any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured because all nSTATUS and CONF_DONE pins are tied together. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

You can have multiple devices that contain the same configuration data in your system. To support this configuration scheme, all device nCE inputs are tied to GND, while the nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[0], and CONF_DONE) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered. Devices must be of the same density and package. All devices start and complete configuration at the same time.

Figure 9–16 shows a multi-device PS configuration when both Cyclone III device family is receiving the same configuration data.

Figure 9–16. Multi-Device PS Configuration When Both Devices Receive the Same Data



Notes to Figure 9–16:

(1) The pull-up resistor must be connected to a supply that provides an acceptable input signal for all devices in the chain. V_{CC} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

(2) The `nCEO` pins of both devices are left unconnected or used as user I/O pins when configuring the same configuration data into multiple devices.

(3) The MSEL pin settings vary for different configuration voltage standards and POR time. To connect `MSEL[3..0]`, refer to Table 9–7 on page 9–11. Connect the MSEL pins directly to V_{CCA} or ground.

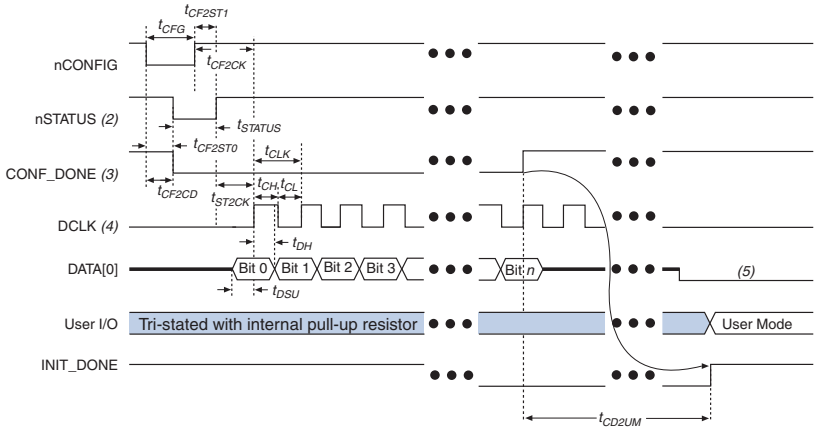
(4) All I/O inputs must maintain a maximum AC voltage of 4.1 V. `DATA[0]` and `DCLK` must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 9–7.

PS Configuration Timing

A PS configuration must meet the setup and hold timing parameters and the maximum clock frequency. When using a microprocessor or another intelligent host to control the PS interface, ensure that you meet these timing requirements.

Figure 9–17 shows the timing waveform for a PS configuration when using an external host device as an external host.

Figure 9–17. PS Configuration Timing Waveform *(Note 1)*



Notes to Figure 9–17:

(1) The beginning of this waveform shows the device in user mode. In user mode, nCONFIG, nSTATUS, and CONF_DONE are at logic-high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.

(2) After power-up, the Cyclone III device family holds nSTATUS low during POR delay.

(3) After power-up, before and during configuration, CONF_DONE is low.

(4) In user mode, drive DCLK either high or low when using the PS configuration scheme, whichever is more convenient. When using the AS configuration scheme, DCLK is a Cyclone III device family output pin and must not be driven externally.

(5) Do not leave the DATA[0] pin floating after configuration. Drive it high or low, whichever is more convenient.

Table 9–13 lists the PS configuration timing parameters for Cyclone III device family.

Table 9–13. PS Configuration Timing Parameters for Cyclone III Device Family *(Note 1)* (Part 1 of 2)

Symbol	Parameter	Minimum	Maximum	Unit
t_{CF2CD}	nCONFIG low to CONF_DONE low	—	500	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low	—	500	ns
t_{CFG}	nCONFIG low pulse width	500	—	ns
t_{STATUS}	nSTATUS low pulse width	45	230 *(2)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high	—	230 *(2)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	230 *(2)*	—	µs
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2	—	µs
t_{DSU}	Data setup time before rising edge on DCLK	5	—	ns
t_{DH}	Data hold time after rising edge on DCLK	0	—	ns
t_{CH}	DCLK high time	3.2	—	ns
t_{CL}	DCLK low time	3.2	—	ns
t_{CLK}	DCLK period	7.5	—	ns

Table 9–13. PS Configuration Timing Parameters for Cyclone III Device Family *(Note 1)* (Part 2 of 2)

Symbol	Parameter	Minimum	Maximum	Unit
f_{MAX}	DCLK frequency	—	100 *(4)*	MHz
t_{CD2UM}	CONF_DONE high to user mode *(3)*	300	650	µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period	—	—
t_{CD2UMC}	CONF_DONE high to user mode with **CLKUSR** option on	t_{CD2CU} + (initialization clock cycles × CLKUSR period) *(5)*	—	—

Notes to Table 9–13:

(1) This information is preliminary.
(2) This value is applicable if you do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.
(3) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting the device.
(4) Cyclone III devices can support a DCLK f_{MAX} of 133 MHz. Cyclone III LS devices can support a DCLK f_{MAX} of 100 MHz.
(5) For more information about the initialization clock cycles required in Cyclone III device family, refer to Table 9–5 on page 9–10.

PS Configuration Using a Download Cable

In this section, the generic term "download cable" includes the Altera USB-Blaster universal serial bus (USB) port download cable, MasterBlaster™ serial/USB communications cable, ByteBlaster II parallel port download cable, the ByteBlasterMV™ parallel port download cable, and the Ethernet-Blaster communications cable.

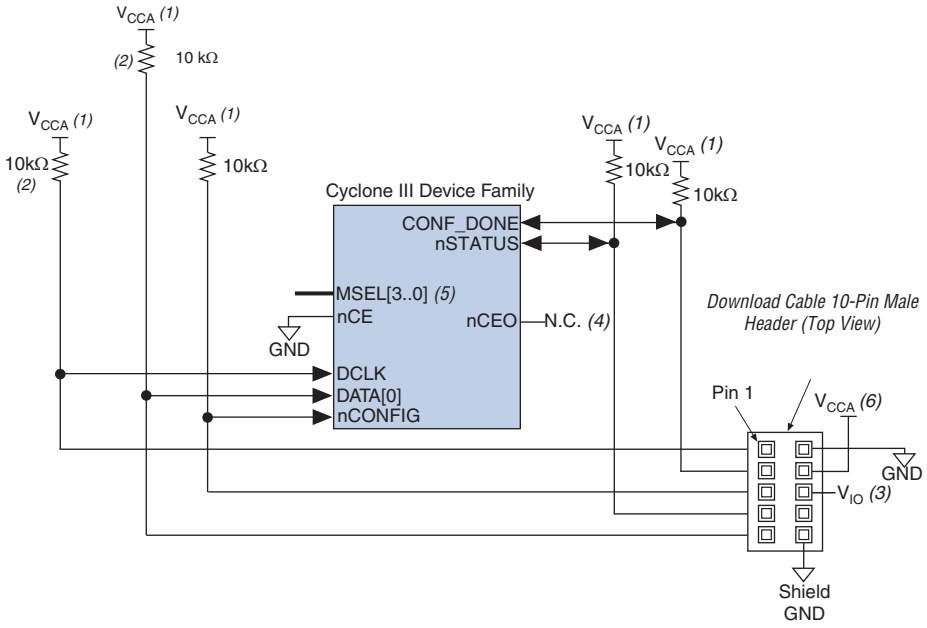
In the PS configuration with a download cable, an intelligent host (such as a PC) transfers data from a storage device to the device using the download cable.

The programming hardware or download cable then places the configuration data one bit at a time on the DATA[0] pin of the device. The configuration data is clocked into the target device until CONF_DONE goes high. The CONF_DONE pin must have an external 10-kΩ pull-up resistor for the device to initialize.

When you use a download cable, setting the **Auto-restart configuration after error** option does not affect the configuration cycle because you must manually restart configuration in the Quartus II software when an error occurs. Additionally, the **Enable user-supplied start-up clock (CLKUSR)** option has no effect on the device initialization because this option is disabled in the **.sof** when programming the device using the Quartus II programmer and download cable. Therefore, if you turn on the **CLKUSR** option, you do not need to provide a clock on CLKUSR when you are configuring the device with the Quartus II programmer and a download cable.

Figure 9–18 shows PS configuration for Cyclone III device family using a download cable.

Figure 9–18. PS Configuration Using a USB-Blaster, MasterBlaster, ByteBlaster II, ByteBlasterMV, or Ethernet Blaster Cable



Notes to Figure 9–18:

(1) The pull-up resistor must be connected to the same supply voltage as the V_{CCA} supply.

(2) You only need the pull-up resistors on DATA[0] and DCLK if the download cable is the only configuration scheme used on your board. This is to ensure that DATA[0] and DCLK are not left floating after configuration. For example, if you are also using a configuration device, you do not need the pull-up resistors on DATA[0] and DCLK.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} must match the V_{CCA} of the device. For this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*. For the USB Blaster, ByteBlaster II, ByteBlaster MV, and Ethernet Blaster, this pin is a no connect.

(4) The nCEO pin is left unconnected or used as a user I/O pin when it does not feed the nCE pin of another device.

(5) The MSEL pin settings vary for different configuration voltage standards and POR time. To connect MSEL[3..0], refer to Table 9–7 on page 9–11 for PS configuration schemes. Connect the MSEL pins directly to V_{CCA} or GND.

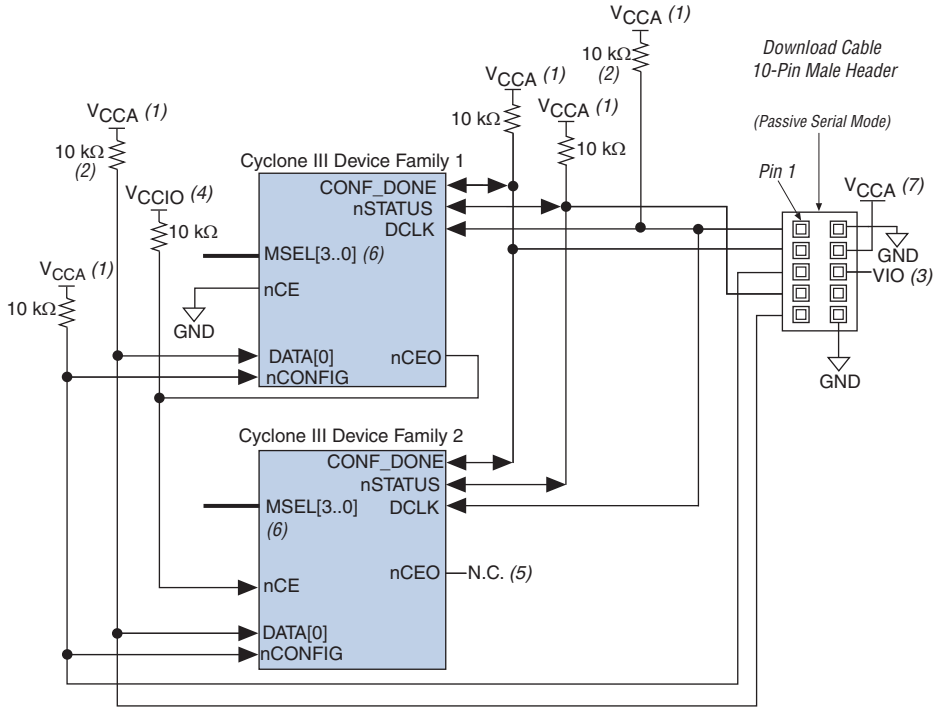
(6) Power up the V_{CC} of the ByteBlaster II, USB-Blaster, or ByteBlasterMV cable with a 2.5- V supply from V_{CCA}. Third-party programmers must switch to 2.5 V. Pin 4 of the header is a V_{CC} power supply for the MasterBlaster cable. The MasterBlaster cable can receive power from either 5.0- or 3.3-V circuit boards, DC power supply, or 5.0 V from the USB cable. For this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*.

You can use a download cable to configure multiple Cyclone III device family by connecting the nCEO pin of each device to the nCE pin of the subsequent device. The nCE pin of the first device is connected to GND while its nCEO pin is connected to the nCE pin of the next device in the chain. The nCE input of the last device comes from the previous device while its nCEO pin is left floating. All other configuration pins, nCONFIG, nSTATUS, DCLK, DATA[0], and CONF_DONE are connected to every device in the chain. Because all CONF_DONE pins are tied together, all devices in the chain initialize and enter user mode at the same time.

In addition, the entire chain halts configuration if any device detects an error because the nSTATUS pins are tied together. The **Auto-restart configuration after error** option does not affect the configuration cycle because you must manually restart configuration in the Quartus II software when an error occurs.

Figure 9–19 shows PS configuration for multi Cyclone III device family using a MasterBlaster, USB-Blaster, ByteBlaster II, or ByteBlasterMV cable.

Figure 9–19. Multi-Device PS Configuration Using a USB-Blaster, MasterBlaster, ByteBlaster II, ByteBlasterMV, or Ethernet Blaster Cable



Notes to Figure 9–19:

(1) The pull-up resistor must be connected to the same supply voltage as the V_{CCA} supply.

(2) You only need the pull-up resistors on DATA[0] and DCLK if the download cable is the only configuration scheme used on your board. This is to ensure that DATA[0] and DCLK are not left floating after configuration. For example, if you are also using a configuration device, you do not need the pull-up resistors on DATA[0] and DCLK.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} must match the V_{CCA} of the device. For this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*. In ByteBlasterMV, this pin is a no connect. In USB-Blaster, ByteBlaster II, and Ethernet Blaster, this pin is connected to nCE when it is used for AS programming. Otherwise, it is a no connect.

(4) Connect the pull-up resistor to the V_{CCIO} supply voltage of the I/O bank in which the nCE pin resides.

(5) The nCEO pin of the last device in the chain is left unconnected or used as a user I/O pin.

(6) The MSEL pin settings vary for different configuration voltage standards and POR time. To connect MSEL[3..0] for PS configuration schemes, refer to Table 9–7 on page 9–11. Connect the MSEL pins directly to V_{CCA} or GND.

(7) Power up the V_{CC} of the ByteBlaster II, USB-Blaster, or ByteBlasterMV cable with a 2.5- V supply from V_{CCA}. Third-party programmers must switch to 2.5 V. Pin 4 of the header is a V_{CC} power supply for the MasterBlaster cable. The MasterBlaster cable can receive power from either 5.0- or 3.3-V circuit boards, DC power supply, or 5.0 V from the USB cable. For this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*.

FPP Configuration

The FPP configuration in Cyclone III device family is designed to meet the increasing demand for faster configuration time. Cyclone III device family is designed with the capability of receiving byte-wide configuration data per clock cycle.

You can perform the FPP configuration of Cyclone III device family with an intelligent host, such as a MAX II device or microprocessor with flash memory. If your system already contains a CFI flash memory, you can use it for the Cyclone III device family configuration storage as well. The MAX II PFL feature in MAX II devices provides an efficient method to program CFI flash memory devices through the JTAG interface and the logic to control configuration from the flash memory device to the Cyclone III device family. Both PS and FPP configuration schemes are supported using this PFL feature.

For more information about the PFL, refer to *AN 386: Using the Parallel Flash Loader with the Quartus II Software*.

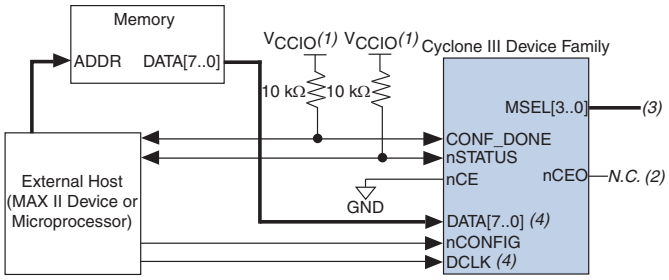
Cyclone III device family does not support enhanced configuration devices for PS or FPP configurations.

FPP configuration is not supported in the E144 package of Cyclone III devices.

FPP Configuration Using an External Host

The FPP configuration using an external host provides a fast method to configure Cyclone III device family. In the FPP configuration scheme, you can use an external host device to control the transfer of configuration data from a storage device, such as flash memory, to the target Cyclone III device family. You can store configuration data in either an **.rbf**, **.hex**, or **.ttf** format. When using the external host, a design that controls the configuration process, such as fetching the data from flash memory and sending it to the device, must be stored in the external host device. Figure 9–20 shows the configuration interface connections between the Cyclone III device family and an external device for single-device configuration.

Figure 9–20. Single-Device FPP Configuration Using an External Host



Notes to Figure 9–20:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for the device. V_{CC} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

(2) The `nCEO` pin is left unconnected or used as a user I/O pin when it does not feed the `nCE` pin of another device.

(3) The MSEL pin settings vary for different configuration voltage standards and POR time. To connect `MSEL[3..0]`, refer to Table 9–7 on page 9–11. Connect the MSEL pins directly to V_{CCA} or GND.

(4) All I/O inputs must maintain a maximum AC voltage of 4.1 V. `DATA[7..0]` and `DCLK` must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 9–7.

After nSTATUS is released, the device is ready to receive configuration data and the configuration stage begins. When nSTATUS is pulled high, the external host device places the configuration data one byte at a time on the DATA[7..0] pins.

Cyclone III device family receives configuration data on the DATA[7..0] pins and the clock is received on the DCLK pin. Data is latched into the device on the rising edge of DCLK. Data is continuously clocked into the target device until CONF_DONE goes high. The CONF_DONE pin goes high one byte early in FPP configuration mode. The last byte is required for serial configuration (AS and PS) modes.

Two DCLK falling edges are required after CONF_DONE goes high to begin the initialization of the device.

Supplying a clock on CLKUSR does not affect the configuration process. After the CONF_DONE pin goes high, CLKUSR is enabled after the time specified as t_{CD2CU}. After this time period elapses, Cyclone III device family requires certain amount of clock cycles to initialize properly and enter user mode. For more information about the initialization clock cycles required in the Cyclone III device family, refer to Table 9–5 on page 9–10. For more information about the supported CLKUSR f_{MAX} value for Cyclone III device family, refer to Table 9–14 on page 9–47.

The INIT_DONE pin is released and pulled high when initialization is complete. The external host device must be able to detect this low-to-high transition which signals the device has entered user mode. When initialization is complete, the device enters user mode. In user mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

To ensure that DCLK and DATA[0] are not left floating at the end of the configuration, the MAX II device must drive them either high or low, whichever is convenient on your board. The DATA[0] pin is available as a user I/O pin after configuration. When you choose the FPP scheme in the Quartus II software, the DATA[0] pin is tri-stated by default in user mode and must be driven by the external host device. To change this default option in the Quartus II software, select the **Dual-Purpose Pins** tab of the **Device and Pin Options** dialog box.

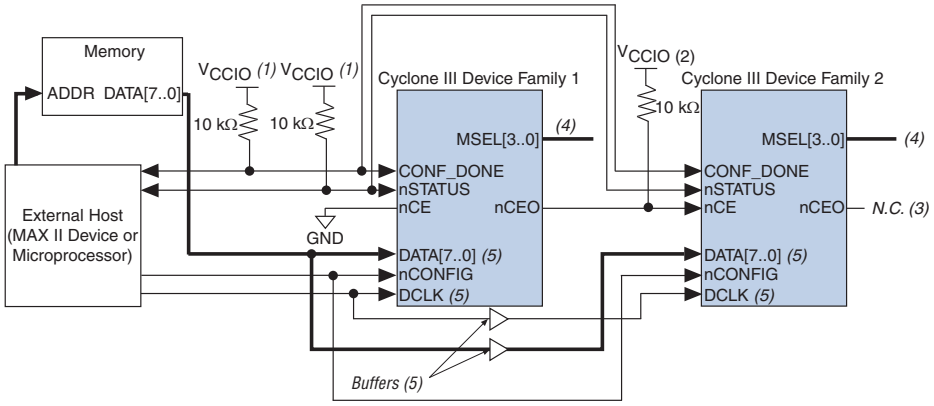
The DCLK speed must be below the specified system frequency to ensure correct configuration. No maximum DCLK period exists, which means you can pause configuration by halting DCLK for an indefinite amount of time.

If a configuration error occurs during configuration and the **Auto-restart configuration after error** option is turned on, the Cyclone III device family releases nSTATUS after a reset time-out period (a maximum of 230 µs). After nSTATUS is released and pulled high by a pull-up resistor, the external host device can try to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the external host device must generate a low-to-high transition (with a low pulse of at least 500 ns) on nCONFIG to restart the configuration process.

The external host device can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The CONF_DONE pin must be monitored by the external device to detect errors and to determine when programming is complete. If all configuration data is sent but CONF_DONE or INIT_DONE has not gone high, the external device must reconfigure the target device.

Figure 9–21 shows how to configure multiple devices using a MAX II device. This circuit is similar to the FPP configuration circuit for a single device, except the Cyclone III device family is cascaded for a multi-device configuration.

Figure 9–21. Multi-Device FPP Configuration Using an External Host



Notes to Figure 9–21:

(1) The pull-up resistor must be connected to a supply that provides an acceptable input signal for all devices in the chain. V_{CC} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

(2) Connect the pull-up resistor to the V_{CCIO} supply voltage of the I/O bank in which the nCE pin resides.

(3) The nCEO pin is left unconnected or used as a user I/O pin when it does not feed the nCE pin of another device.

(4) The MSEL pin settings vary for different configuration voltage standards and POR time. To connect MSEL[3..0], refer to Table 9–7 on page 9–11. Connect the MSEL pins directly to V_{CCA} or ground.

(5) All I/O inputs must maintain a maximum AC voltage of 4.1 V. DATA[7..0] and DCLK must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 9–7.

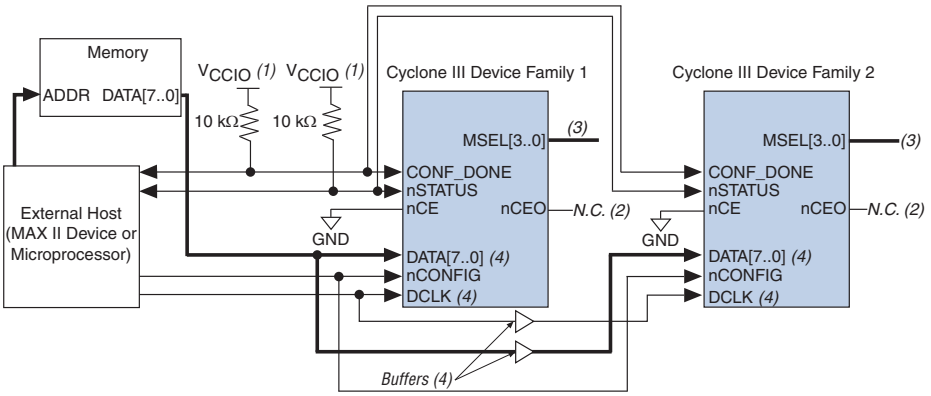
In a multi-device FPP configuration, the nCE pin of the first device is connected to GND while its nCEO pin is connected to the nCE pin of the next device in the chain. The nCE input of the last device comes from the previous device while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the nCE pin of the second device, which prompts the second device to begin configuration. The second device in the chain begins configuration in one clock cycle; therefore, the transfer of data destinations is transparent to the MAX II device. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. The configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered. All devices initialize and enter user mode at the same time because all device CONF_DONE pins are tied together.

All nSTATUS and CONF_DONE pins are tied together and if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

If a system has multiple devices that contain the same configuration data, tie all device nCE inputs to GND and leave nCEO pins floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered. Devices must be of the same density and package. All devices start and complete configuration at the same time.

Figure 9–22 shows multi-device FPP configuration when both Cyclone III device family is receiving the same configuration data.

Figure 9–22. Multi-Device FPP Configuration Using an External Host When Both Devices Receive the Same Data



Notes to Figure 9–22:

(1) The pull-up resistor must be connected to a supply that provides an acceptable input signal for all devices in the chain. V_{CC} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

(2) The nCEO pins of both devices are left unconnected or used as user I/O pins when configuring the same configuration data into multiple devices.

(3) The MSEL pin settings vary for different configuration voltage standards and POR time. To connect MSEL[3..0], refer to Table 9–7 on page 9–11. Connect the MSEL pins directly to V_{CCA} or GND.

(4) All I/O inputs must maintain a maximum AC voltage of 4.1 V. DATA[7..0] and DCLK must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 9–7.

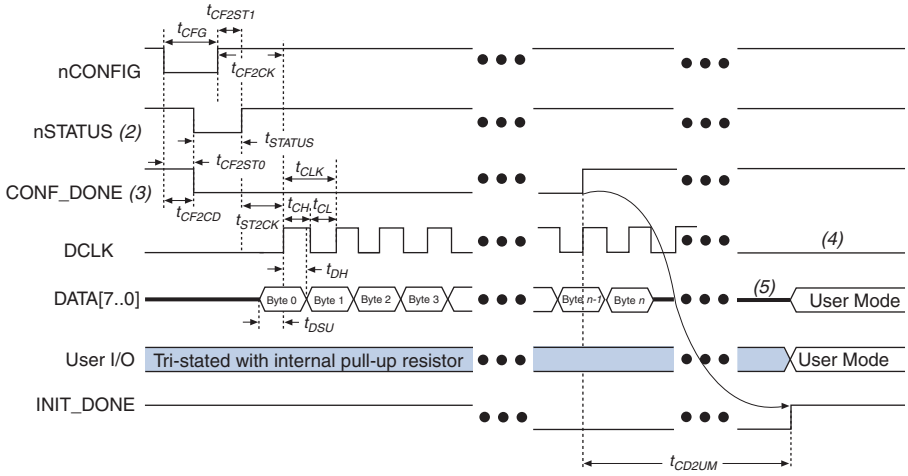
You can use a single configuration chain to configure Cyclone III device family with other Altera devices that support the FPP configuration. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device starts reconfiguration in all devices, tie all the device CONF_DONE and nSTATUS pins together.

For more information about configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook*.

FPP Configuration Timing

Figure 9–23 shows the timing waveform for FPP configuration when using an external host.

Figure 9–23. FPP Configuration Timing Waveform *(Note 1)*



Notes to Figure 9–23:

(1) The beginning of this waveform shows the device in user mode. In user mode, nCONFIG, nSTATUS, and CONF_DONE are at logic-high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.

(2) After power-up, the Cyclone III device family holds nSTATUS low during POR delay.

(3) After power-up, before and during configuration, CONF_DONE is low.

(4) Do not leave DCLK floating after configuration. It must be driven high or low, whichever is more convenient.

(5) DATA[7..0] is available as user I/O pin after configuration; the state of the pin depends on the dual-purpose pin settings.

Table 9–14 lists the FPP configuration timing parameters for Cyclone III device family.

Table 9–14. FPP Timing Parameters for Cyclone III Device Family *(Note 1)* (Part 1 of 2)

Symbol	Parameter	Minimum	Maximum	Unit
t_{CF2CD}	nCONFIG low to CONF_DONE low	—	500	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low	—	500	ns
t_{CFG}	nCONFIG low pulse width	500	—	ns
t_{STATUS}	nSTATUS low pulse width	45	230 *(2)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high	—	230 *(2)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	230 *(2)*	—	µs
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2	—	µs
t_{DSU}	DATA setup time before rising edge on DCLK	5	—	ns
t_{DH}	DATA hold time after rising edge on DCLK	0	—	ns
t_{CH}	DCLK high time	3.2	—	ns
t_{CL}	DCLK low time	3.2	—	ns
t_{CLK}	DCLK period	7.5	—	ns
f_{MAX}	DCLK frequency	—	100 *(4)*	MHz
t_{CD2UM}	CONF_DONE high to user mode *(3)*	300	650	µs

Table 9–14. FPP Timing Parameters for Cyclone III Device Family *(Note 1)* (Part 2 of 2)

Symbol	Parameter	Minimum	Maximum	Unit
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period	—	—
t_{CD2UMC}	CONF_DONE high to user mode with **CLKUSR** option on	t_{CD2CU} + (initialization clock cycles × CLKUSR period) *(5)*	—	—

Notes to Table 9–14:

(1) This information is preliminary.

(2) This value is applicable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(3) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device.

(4) Cyclone III EP3C5, EP3C10, EP3C16, EP3C25, and EP3C40 devices support a DCLK f_{MAX} of 133 MHz. Cyclone III EP3C55, EP3C80, EP3C120 and all the Cyclone III LS devices support a DCLK f_{MAX} of 100 MHz.

(5) For more information about the initialization clock cycles required in Cyclone III device family, refer to Table 9–5 on page 9–10.

JTAG Configuration

JTAG has developed a specification for boundary-scan testing. This boundary-scan test (BST) architecture offers the capability to efficiently test components on PCBs with tight lead spacing. The BST architecture can test pin connections without using physical test probes and capture functional data while a device is operating normally. You can also use the JTAG circuitry to shift configuration data into the device. The Quartus II software automatically generates **.sof**s that are used for JTAG configuration with a download cable in the Quartus II software programmer.

For more information about JTAG boundary-scan testing, refer to the *IEEE 1149.1 (JTAG) Boundary-Scan Testing for Cyclone III Devices* chapter.

JTAG instructions have precedence over any other device configuration modes. Therefore, JTAG configuration can take place without waiting for other configuration modes to complete. For example, if you attempt JTAG configuration of Cyclone III device family during PS configuration, PS configuration terminates and JTAG configuration begins. If the Cyclone III device family MSEL pins are set to AS mode, the Cyclone III device family does not output a DCLK signal when JTAG configuration takes place.

The four required pins for a device operating in JTAG mode are TDI, TDO, TMS, and TCK. The TCK pin has an internal weak pull-down resistor while the TDI and TMS pins have weak internal pull-up resistors (typically 25 kΩ). The TDO output pin is powered by V_{CCIO} in I/O bank 1. All the JTAG input pins are powered by the V_{CCIO} pin. All the JTAG pins support only LVTTL I/O standard. All user I/O pins are tri-stated during JTAG configuration. Table 9–15 lists the function of each JTAG pin.

The TDO output is powered by the V_{CCIO} power supply of I/O bank 1.

For more information about how to connect a JTAG chain with multiple voltages across the devices in the chain, refer to the *IEEE 1149.1 (JTAG) Boundary-Scan Testing for Cyclone III Devices* chapter.

Table 9–15. Dedicated JTAG Pins

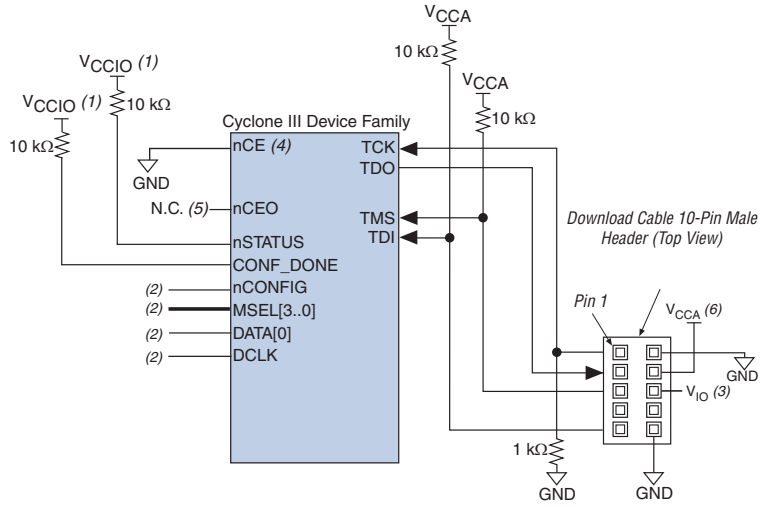
Pin Name	Pin Type	Description
TDI	Test data input	Serial input pin for instructions as well as test and programming data. Data shifts in on the rising edge of TCK. The TDI pin is powered by the V_{CCIO} supply. If the JTAG interface is not required on the board, the JTAG circuitry is disabled by connecting this pin to V_{CC}.
TDO	Test data output	Serial data output pin for instructions as well as test and programming data. Data shifts out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. The TDO pin is powered by V_{CCIO} in I/O bank 1. If the JTAG interface is not required on the board, the JTAG circuitry is disabled by leaving this pin unconnected.
TMS	Test mode select	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions in the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. The TMS pin is powered by the V_{CCIO} supply. If the JTAG interface is not required on the board, the JTAG circuitry is disabled by connecting this pin to V_{CC}.
TCK	Test clock input	Clock input to the BST circuitry. Some operations occur at the rising edge while others occur at the falling edge. The TCK pin is powered by the V_{CCIO} supply. If the JTAG interface is not required on the board, the JTAG circuitry is disabled by connecting this pin to GND.

You can download data to the device on the PCB through the USB-Blaster, MasterBlaster, ByteBlaster II, ByteBlasterMV download cable, and Ethernet-Blaster communications cable during JTAG configuration. Configuring devices using a cable is similar to programming devices in-system. Figure 9–24 and Figure 9–25 show the JTAG configuration of a single Cyclone III device family.

For device V_{CCIO} of 2.5, 3.0, and 3.3 V, refer to Figure 9–24. All I/O inputs must maintain a maximum AC voltage of 4.1 V. Because JTAG pins do not have the internal PCI clamping diodes to prevent voltage overshoot when using V_{CCIO} of 2.5, 3.0, and 3.3 V, you must power up the V_{CC} of the download cable with a 2.5-V supply from V_{CCA}.

For device V_{CCIO} of 1.2, 1.5, and 1.8 V, refer to Figure 9–25. You can power up the V_{CC} of the download cabled with the supply from V_{CCIO}.

Figure 9–24. JTAG Configuration of a Single Device Using a Download Cable (2.5, 3.0, and 3.3-V V_{CCIO} Powering the JTAG Pins)



Notes to Figure 9–24:

(1) Connect these pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) Connect the `nCONFIG` and `MSEL[3..0]` pins to support a non-JTAG configuration scheme. If you only use a JTAG configuration, connect the `nCONFIG` pin to logic high and the `MSEL[3..0]` pins to ground. In addition, pull `DCLK` and `DATA[0]` either high or low, whichever is convenient on your board.

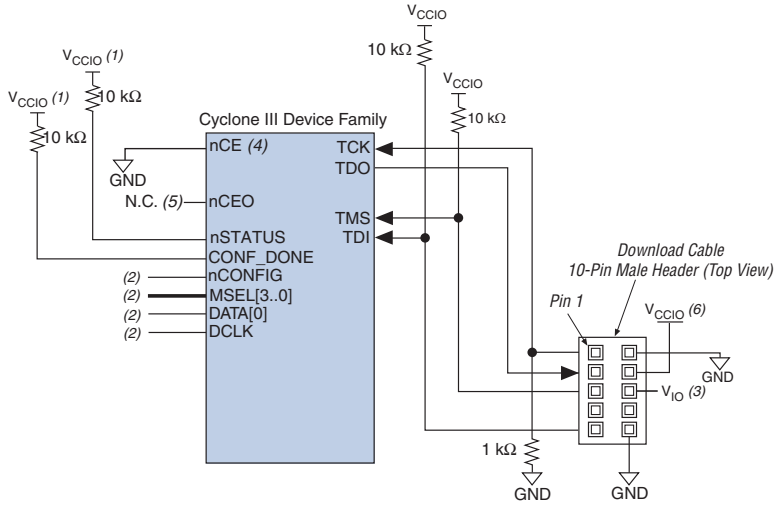
(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} must match the device's V_{CCA}. For this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*. In USB-Blaster, ByteBlaster II, ByteBlasterMV, and Ethernet Blaster, this pin is a no connect.

(4) The `nCE` pin must be connected to GND or driven low for successful JTAG configuration.

(5) The `nCEO` pin is left unconnected or used as a user I/O pin when it does not feed the `nCE` pin of another device.

(6) Power up the V_{CC} of the ByteBlaster II, USB-Blaster, ByteBlasterMV, or Ethernet Blaster cable with a 2.5- V supply from V_{CCA}. Third-party programmers must switch to 2.5 V. Pin 4 of the header is a V_{CC} power supply for the MasterBlaster cable. The MasterBlaster cable can receive power from either 5.0- or 3.3-V circuit boards, DC power supply, or 5.0 V from the USB cable. For this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*.

Figure 9–25. JTAG Configuration of a Single Device Using a Download Cable (1.5-V or 1.8-V V_{CCIO} Powering the JTAG Pins)



Notes to Figure 9–25:

(1) Connect these pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) Connect the nCONFIG and MSEL[3..0] pins to support a non-JTAG configuration scheme. If you only use a JTAG configuration, connect the nCONFIG pin to logic-high and the MSEL[3..0] pins to ground. In addition, pull DCLK and DATA[0] either high or low, whichever is convenient on your board.

(3) In the USB-Blaster and ByteBlaster II cables, this pin is connected to nCE when it is used for AS programming; otherwise it is a no connect.

(4) The nCE must be connected to GND or driven low for successful JTAG configuration.

(5) The nCEO pin is left unconnected or used as a user I/O pin when it does not feed the nCE pin of another device.

(6) Power up the V_{CC} of the ByteBlaster II, USB-Blaster, or Ethernet Blaster cable with supply from V_{CCIO}. The ByteBlaster II, USB-Blaster, and Ethernet Blaster cables do not support a target supply voltage of 1.2 V. For the target supply voltage value, refer to the *ByteBlaster II Download Cable User Guide*, *USB-Blaster Download Cable User Guide* and *Ethernet Blaster Communications Cable User Guide*.

To configure a single device in a JTAG chain, the programming software places all other devices in bypass mode. In bypass mode, devices pass programming data from the TDI pin to the TDO pin through a single bypass register without being affected internally. This scheme enables the programming software to program or verify the target device. Configuration data driven into the device appears on the TDO pin one clock cycle later.

The Quartus II software verifies successful JTAG configuration upon completion. At the end of configuration, the software checks the state of CONF_DONE through the JTAG port. When the Quartus II software generates a **.jam** for a multi-device chain, it contains instructions to have all devices in the chain initialize at the same time. If CONF_DONE is not high, the Quartus II software indicates that configuration has failed. If CONF_DONE is high, the software indicates that configuration was successful. After the configuration bitstream is serially sent using the JTAG TDI port, the TCK port clocks an additional clock cycle to perform device initialization.

Cyclone III device family has dedicated JTAG pins that function as JTAG pins. You can perform JTAG testing on Cyclone III device family before, during, and after configuration. Cyclone III device family supports the BYPASS, IDCODE, and SAMPLE instructions during configuration without interrupting configuration. All other JTAG instructions can only be issued by first interrupting configuration and reprogramming I/O pins using the ACTIVE_DISENGAGE and CONFIG_IO instructions.

The CONFIG_IO instruction allows I/O buffers to be configured using the JTAG port and when issued after the ACTIVE_DISENGAGE instruction interrupts configuration. This instruction allows you to perform board-level testing prior to configuring the Cyclone III device family or waiting for a configuration device to complete configuration. Prior to issuing the CONFIG_IO instruction, you must issue the ACTIVE_DISENGAGE instruction. This is because in Cyclone III device family, the CONFIG_IO instruction does not hold nSTATUS low until reconfiguration, so you must disengage the active configuration mode controller when active configuration is interrupted. The ACTIVE_DISENGAGE instruction places the active configuration mode controllers in an idle state prior to JTAG programming. Additionally, the ACTIVE_ENGAGE instruction allows you to re-engage a disengaged active configuration mode controller.

You must follow a specific flow when executing the CONFIG_IO, ACTIVE_DISENGAGE, and ACTIVE_ENGAGE JTAG instructions in Cyclone III device family. For more information about the instruction flow, refer to "JTAG Instructions" on page 9–61.

The chip-wide reset (DEV_CLRn) and chip-wide output enable (DEV_OE) pins on Cyclone III device family do not affect JTAG boundary-scan or programming operations. Toggling these pins does not affect JTAG operations (other than the usual boundary-scan operation).

When designing a board for JTAG configuration, consider the dedicated configuration pins. Table 9–16 lists how these pins must be connected during JTAG configuration.

Table 9–16. Dedicated Configuration Pin Connections During JTAG Configuration

Signal	Description
nCE	On all Cyclone III device family in the chain, nCE must be driven low by connecting it to ground, pulling it low using a resistor or driving it by some control circuitry. For devices that are also in multi-device AS, AP, PS, or FPP configuration chains, the nCE pins must be connected to GND during JTAG configuration or JTAG configured in the same order as the configuration chain.
nCEO	On all Cyclone III device family in the chain, nCEO is left floating or connected to the nCE of the next device.
MSEL[3..0]	These pins must not be left floating. These pins support whichever non-JTAG configuration that is used in production. If you only use JTAG configuration, tie these pins to GND.
nCONFIG	Driven high by connecting to the V_{CCIO} supply of the bank in which the pin resides and pulling up using a resistor or driven high by some control circuitry.
nSTATUS	Pull to the V_{CCIO} supply of the bank in which the pin resides using a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each nSTATUS pin must be pulled up to the V_{CCIO} individually.
CONF_DONE	Pull to the V_{CCIO} supply of the bank in which the pin resides using a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each CONF_DONE pin must be pulled up to the V_{CCIO} supply of the bank in which the pin resides individually. CONF_DONE going high at the end of JTAG configuration indicates successful configuration.
DCLK	Must not be left floating. Drive low or high, whichever is more convenient on your board.

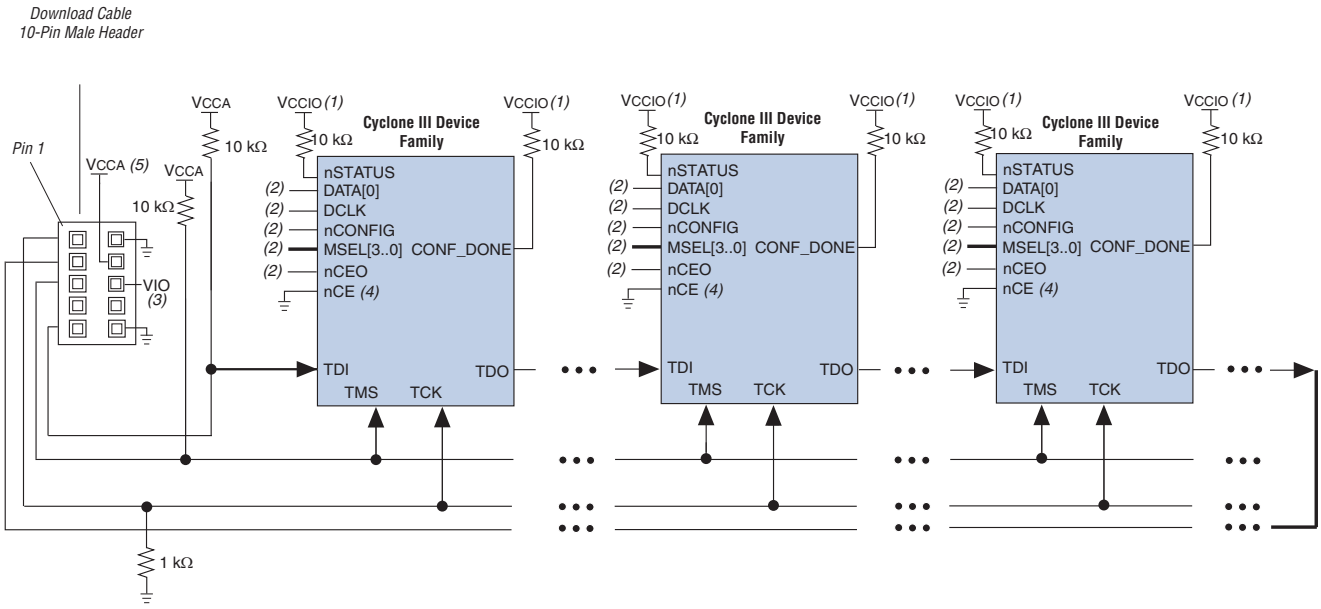
When programming a JTAG device chain, one JTAG-compatible header is connected to several devices. The number of devices in the JTAG chain is limited only by the drive capability of the download cable. When four or more devices are connected in a JTAG chain, Altera recommends buffering the TCK, TDI, and TMS pins with an on-board buffer.

JTAG-chain device programming is ideal when the system contains multiple devices, or when testing your system using JTAG BST circuitry. Figure 9–26 and Figure 9–27 show a multi-device JTAG configuration.

For the device V_{CCIO} of 2.5, 3.0, and 3.3 V, refer to Figure 9–26. All I/O inputs must maintain a maximum AC voltage of 4.1 V. Because JTAG pins do not have the internal PCI clamping diodes to prevent voltage overshoot when using V_{CCIO} of 2.5, 3.0, and 3.3 V, you must power up the V_{CC} of the download cable with a 2.5-V supply from V_{CCA}.

For device V_{CCIO} of 1.2, 1.5, and 1.8 V, refer to Figure 9–27. You can power up the V_{CC} of the download cable with the supply from V_{CCIO}.

Figure 9–26. JTAG Configuration of Multiple Devices Using a Download Cable (2.5, 3.0, and 3.3-V V_{CCIO} Powering the JTAG Pins)



Notes to Figure 9–26:

(1) Connect these pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

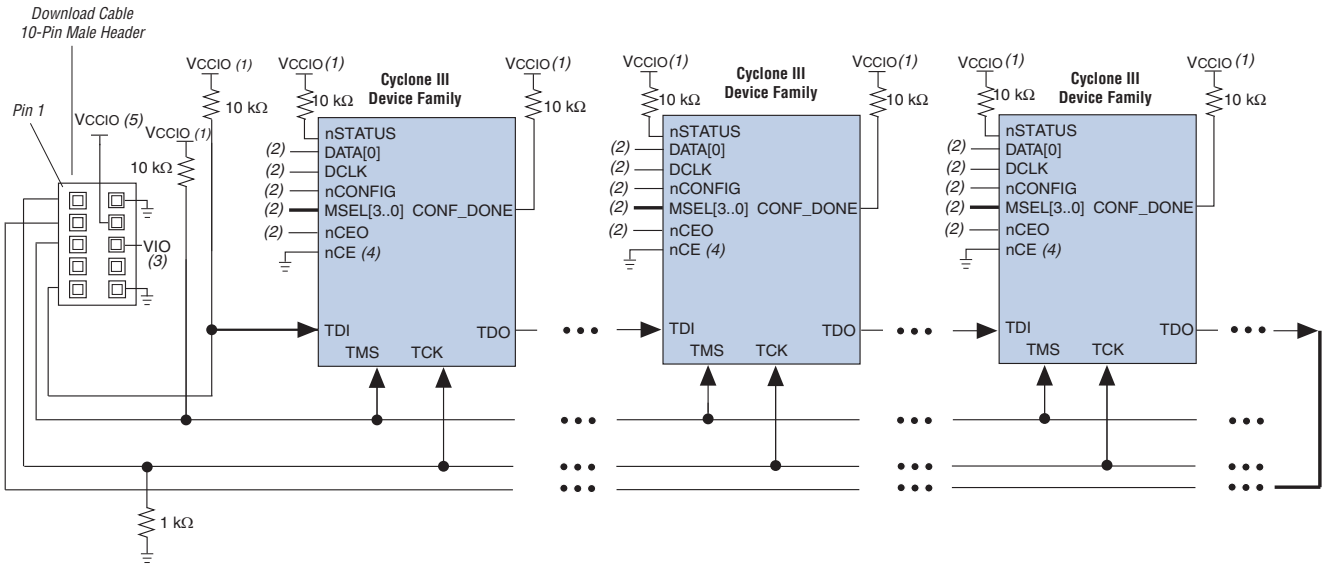
(2) Connect the `nCONFIG` and `MSEL[3..0]` pins to support a non-JTAG configuration scheme. If you only use a JTAG configuration, connect the `nCONFIG` pin to logic high and the `MSEL[3..0]` pins to ground. In addition, pull `DCLK` and `DATA[0]` either high or low, whichever is convenient on your board.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} must match the V_{CCA} of the device. For this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*. In the ByteBlasterMV cable, this pin is a no connect. In the USB-Blaster and ByteBlaster II cables, this pin is connected to `nCE` when it is used for AS programming, otherwise it is a no connect.

(4) The `nCE` pin must be connected to ground or driven low for successful JTAG configuration.

(5) Power up the V_{CC} of the ByteBlaster II, USB-Blaster, or ByteBlasterMV cable with a 2.5- V supply from V_{CCA}. Third-party programmers must switch to 2.5 V. Pin 4 of the header is a V_{CC} power supply for the MasterBlaster cable. The MasterBlaster cable can receive power from either 5.0- or 3.3-V circuit boards, DC power supply, or 5.0 V from the USB cable. For this value, refer to the *MasterBlaster Serial/USB Communications User Guide*.

Figure 9–27. JTAG Configuration of Multiple Devices Using a Download Cable (1.2, 1.5, and 1.8-V V_{CCIO} Powering the JTAG Pins)



Notes to Figure 9–27:

(1) Connect these pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) Connect the nCONFIG and MSEL[3..0] pins to support a non-JTAG configuration scheme. If you only use a JTAG configuration, connect the nCONFIG pin to logic high and the MSEL[3..0] pins to ground. In addition, pull DCLK and DATA[0] either high or low, whichever is convenient on your board.

(3) In the USB-Blaster and ByteBlaster II cable, this pin is connected to nCE when it is used for AS programming, otherwise it is a no connect.

(4) The nCE pin must be connected to ground or driven low for successful JTAG configuration.

(5) Power up the V_{CC} of the ByteBlaster II or USB-Blaster cable with supply from V_{CCIO}. The ByteBlaster II and USB-Blaster cables do not support a target supply voltage of 1.2 V. For the target supply voltage value, refer to the *ByteBlaster II Download Cable User Guide* and the *USB-Blaster Download Cable User Guide*.

All I/O inputs must maintain a maximum AC voltage of 4.1 V. If a non-Cyclone III device family is cascaded in the JTAG-chain, TDO of the non-Cyclone III device family driving into TDI of the Cyclone III device family must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 9–7.

The nCE pin must be connected to GND or driven low during JTAG configuration. In multi-device AS, AP, PS, and FPP configuration chains, the nCE pin of the first device is connected to GND while its nCEO pin is connected to the nCE pin of the next device in the chain. The inputs of the nCE pin of the last device come from the previous device while its nCEO pin is left floating. In addition, the CONF_DONE and nSTATUS signals are shared in multi-device AS, AP, PS, and FPP configuration chains to ensure that the devices enter user mode at the same time after configuration is complete. When the CONF_DONE and nSTATUS signals are shared among all the devices, every device must be configured when you perform JTAG configuration.

If you only use JTAG configuration, Altera recommends that you connect the circuitry as shown in Figure 9–26 or Figure 9–27, in which each of the CONF_DONE and nSTATUS signals are isolated so that each device can enter user mode individually.

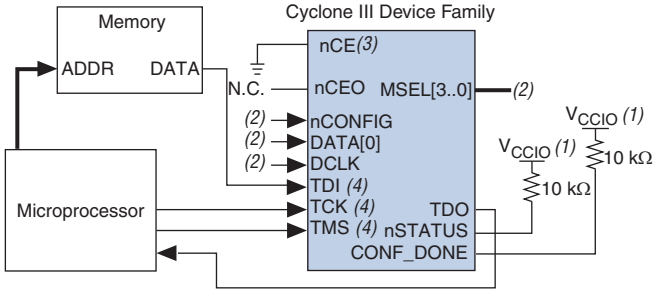
After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the nCE pin of the second device, which prompts the second device to begin configuration. Therefore, if these devices are also in a JTAG chain, ensure that the nCE pins are connected to GND during JTAG configuration or that the devices are JTAG configured in the same order as the configuration chain. As long as the devices are JTAG configured in the same order as the multi-device configuration chain, the nCEO pin of the previous device drives the nCE pin of the next device low when it has successfully been JTAG configured. You can place other Altera devices that have JTAG support in the same JTAG chain for device programming and configuration.

JTAG configuration allows an unlimited number of Cyclone III device family to be cascaded in a JTAG chain.

For more information about configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook*.

Figure 9–28 shows JTAG configuration of a Cyclone III device family with a microprocessor.

Figure 9–28. JTAG Configuration of a Single Device Using a Microprocessor



Notes to Figure 9–28:

(1) The pull-up resistor must be connected to a supply that provides an acceptable input signal for all devices in the chain.

(2) Connect the nCONFIG and MSEL[3..0] pins to support a non-JTAG configuration scheme. If you only use a JTAG configuration, connect the nCONFIG pin to logic high and the MSEL[3..0] pins to ground. In addition, pull DCLK and DATA[0] either high or low, whichever is convenient on your board.

(3) The nCE pin must be connected to GND or driven low for successful JTAG configuration.

(4) All I/O inputs must maintain a maximum AC voltage of 4.1 V. Signals driving into TDI, TMS, and TCK must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 9–7.

Configuring Cyclone III Device Family with Jam STAPL

Jam STAPL, JEDEC standard JESD-71, is a standard file format for in-system programmability (ISP) purposes. Jam STAPL supports programming or configuration of programmable devices and testing of electronic systems, using the IEEE 1149.1 JTAG interface. Jam STAPL is a freely licensed open standard. The Jam Player provides an interface for manipulating the IEEE Std. 1149.1 JTAG TAP state machine.

For more information about JTAG and Jam STAPL in embedded environments, refer to *AN 425: Using Command-Line Jam STAPL Solution for Device Programming*. To download the jam player, visit the Altera website (www.altera.com).

Configuring Cyclone III Device Family with the JRunner Software Driver

The JRunner software driver allows you to configure Cyclone III device family through the ByteBlaster II or ByteBlasterMV cables in JTAG mode. The supported programming input file is in **.rbf** format. The JRunner software driver also requires a Chain Description File (**.cdf**) generated by the Quartus II software. The JRunner software driver is targeted for embedded JTAG configuration. The source code is developed for the Windows NT operating system (OS). You can customize the code to make it run on your embedded platform.

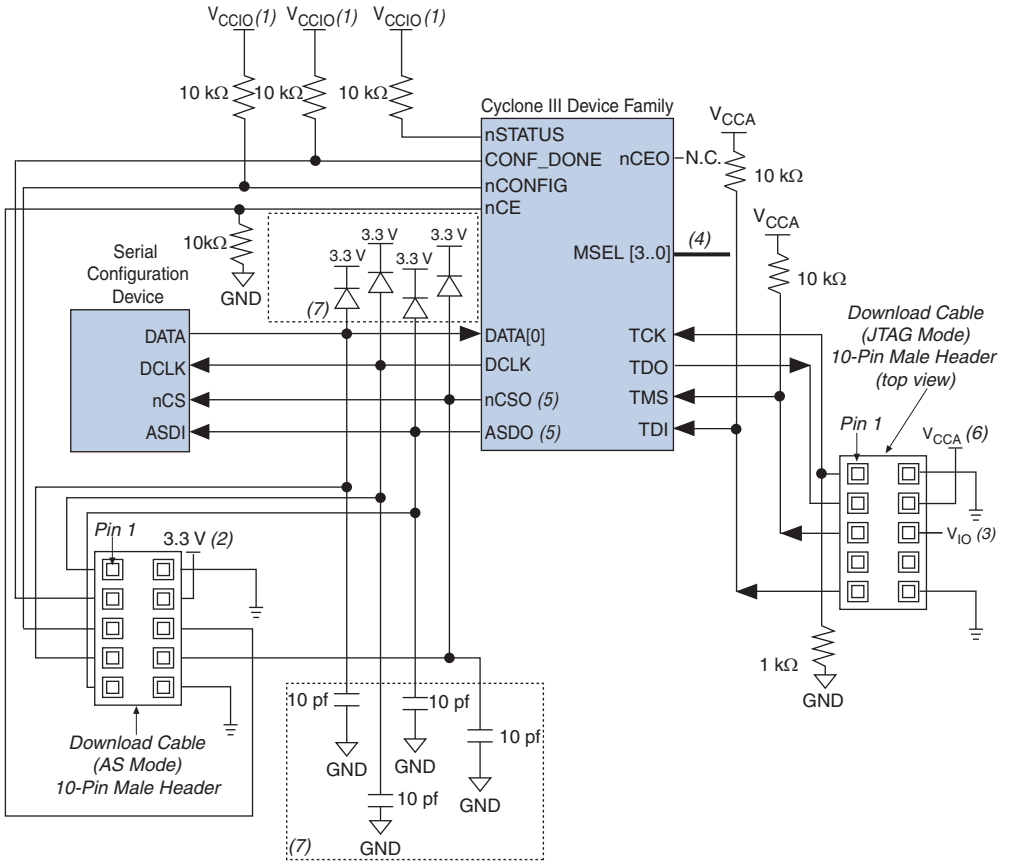
The **.rbf** used by the JRunner software driver cannot be a compressed **.rbf** because the JRunner software driver uses JTAG-based configuration. During JTAG-based configuration, the real-time decompression feature is not available.

For more information about the JRunner software driver, refer to *AN 414: JRunner Software Driver: An Embedded Solution for PLD JTAG Configuration* and the source files on the Altera website at (www.altera.com).

Combining JTAG and AS Configuration Schemes

You can combine the AS configuration scheme with the JTAG-based configuration (Figure 9–29). This setup uses two 10-pin download cable headers on the board. One download cable is used in JTAG mode to configure the Cyclone III device family directly using the JTAG interface. The other download cable is used in AS mode to program the serial configuration device in-system using the AS programming interface. The `MSEL[3..0]` pins must be set to select AS configuration mode (Table 9–7 on page 9–11). If you try configuring the device using both schemes simultaneously, the JTAG configuration takes precedence and the AS configuration terminates.

Figure 9–29. Combining JTAG and AS Configuration Schemes



Notes to Figure 9–29:

(1) Connect these pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) Power up the V_{CC} of the ByteBlaster II, USB-Blaster, or Ethernet Blaster cable with the 3.3-V supply.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} must match the V_{CCA} of the device. For this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*. In ByteBlasterMV, this pin is a no connect. In USB-Blaster and ByteBlaster II, this pin is connected to `nCE` when it is used for AS programming, otherwise it is a no connect.

(4) The MSEL pin settings vary for different configuration voltage standards and POR time. To connect `MSEL[3..0]` for AS configuration schemes, refer to Table 9–7 on page 9–11. Connect the MSEL pins directly to V_{CCA} or GND.

(5) These are dual-purpose I/O pins. This `nCSO` pin functions as the `FLASH_NCE` pin in AP mode. The `ASDO` pin functions as the `DATA[1]` pin in other AP and FPP modes.

(6) Power up V_{CC} of the ByteBlaster II, USB-Blaster, ByteBlasterMV, or Ethernet Blaster cable with a 2.5- V supply from V_{CCA}. Third-party programmers must switch to 2.5 V. Pin 4 of the header is a V_{CC} power supply for the MasterBlaster cable. The MasterBlaster cable can receive power from either 5.0- or 3.3-V circuit boards, DC power supply, or 5.0 V from the USB cable. For this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*.

(7) You must place the diodes and capacitors as close as possible to the Cyclone III device family. For effective voltage clamping, Altera recommends using the Schottky diode, which has a relatively lower forward diode voltage (VF) than the switching and Zener diodes. For more information about the interface guidelines using Schottky diodes, refer to *AN 523: Cyclone III Configuration Interface Guidelines with EPCS Devices.*

Programming Serial Configuration Devices In-System Using the JTAG Interface

Cyclone III device family in a single-device or in a multiple-device chain supports in-system programming of a serial configuration device with the JTAG interface using the SFL design. The intelligent host or download cable of the board can use the four JTAG pins on the Cyclone III device family to program the serial configuration device in system, even if the host or download cable cannot access the configuration pins (DCLK, DATA, ASDI, and nCS pins).

The SFL design is a JTAG-based in-system programming solution for Altera serial configuration devices. The SFL is a bridge design for the Cyclone III device family that uses its JTAG interface to access the EPCS JTAG Indirect Configuration Device Programming (**.jic**) file and then uses the AS interface to program the EPCS device. Both the JTAG interface and AS interface are bridged together inside the SFL design.

In a multiple device chain, you must only configure the master device that controls the serial configuration device. When using this feature, the slave devices in the multiple device chain which are configured by the serial configuration device do not need to be configured. To use this feature successfully, set the MSEL[3..0] pins of the master device to select the AS configuration scheme (Table 9–7 on page 9–11). The serial configuration device in-system programming through the Cyclone III device family JTAG interface has three stages, which are described in the following sections:

- "Loading the SFL Design" on page 9–59
- "ISP of the Configuration Device" on page 9–60
- "Reconfiguration" on page 9–61

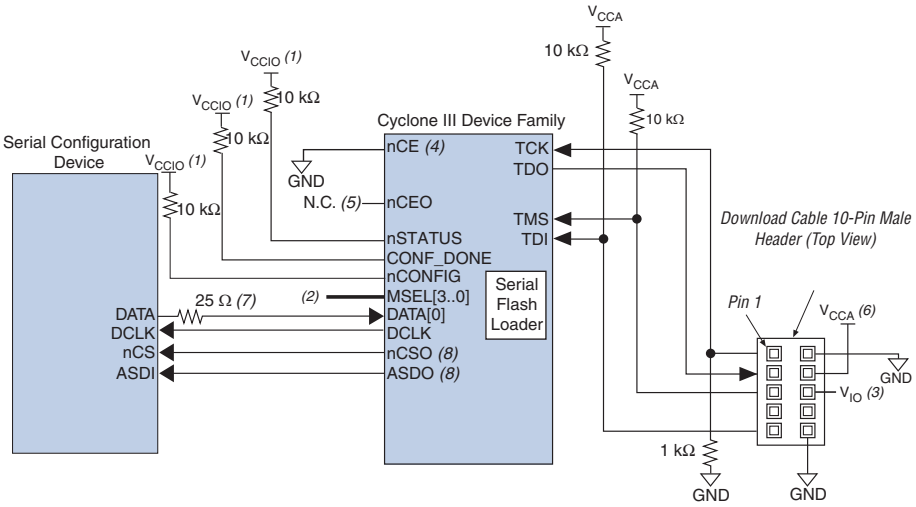
Loading the SFL Design

The SFL design is a design inside the Cyclone III device family that bridges the JTAG interface and the AS interface with glue logic.

The intelligent host uses the JTAG interface to configure the master device with a SFL design. The SFL design allows the master device to control the access of four serial configuration device pins, also known as the Active Serial Memory Interface (ASMI) pins, through the JTAG interface. The ASMI pins are serial clock input (DCLK), serial data output (DATA), AS data input (ASDI), and active-low chip select (nCS) pins.

If you configure a master device with a SFL design, the master device enters user mode even though the slave devices in the multiple device chain are not being configured. The master device enters user mode with a SFL design even though the CONF_DONE signal is externally held low by the other slave devices in chain. Figure 9–30 shows the JTAG configuration of a single Cyclone III device family with a SFL design.

Figure 9–30. Programming Serial Configuration Devices In-System Using the JTAG Interface



Notes to Figure 9–30:

(1) Connect the pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.

(2) The MSEL pin settings vary for different configuration voltage standards and POR time. To connect `MSEL[3..0]` for AS configuration schemes, refer to Table 9–7 on page 9–11. Connect the MSEL pins directly to V_{CCA} or GND.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} must match the V_{CCA} of the device. For this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*. In ByteBlasterMV, this pin is a no connect. In USB-Blaster, ByteBlaster II, and Ethernet Blaster, this pin is connected to `nCE` when it is used for AS programming, otherwise it is a no connect.

(4) The `nCE` pin must be connected to GND or driven low for successful JTAG configuration.

(5) The `nCEO` pin is left unconnected or used as a user I/O pin when it does not feed the `nCE` pin of another device.

(6) Power up the V_{CC} of the ByteBlaster II, USB-Blaster, ByteBlasterMV, or Ethernet Blaster cable with a 2.5-V supply from V_{CCA}. Third-party programmers must switch to 2.5 V. Pin 4 of the header is a V_{CC} power supply for the MasterBlaster cable. The MasterBlaster cable can receive power from either 5.0- or 3.3-V circuit boards, DC power supply, or 5.0 V from the USB cable. For this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*.

(7) Connect the series resistor at the near end of the serial configuration device.

(8) These are dual-purpose I/O pins. The `nCSO` pin functions as the `FLASH_NCE` pin in AP mode. The `ASDO` pin functions as the `DATA[1]` pin in other AP and FPP modes.

ISP of the Configuration Device

In the second stage, the SFL design in the master device allows you to write the configuration data for the device chain into the serial configuration device with the Cyclone III device family JTAG interface. The JTAG interface sends the programming data for the serial configuration device to the Cyclone III device family first. The Cyclone III device family then uses the ASMI pins to send the data to the serial configuration device.

Reconfiguration

After the configuration data is successfully written into the serial configuration device, the Cyclone III device family does not reconfigure by itself. The intelligent host issues the PULSE_NCONFIG JTAG instruction to initialize the reconfiguration process. During reconfiguration, the master device is reset and the SFL design no longer exists in the Cyclone III device family and the serial configuration device configures all the devices in the chain with your user design.

For more information about SFL, refer to *AN 370: Using the Serial FlashLoader with Quartus II Software.*

JTAG Instructions

This section describes the instructions that are necessary for JTAG configuration for the Cyclone III device family. Table 9–17 lists the supported JTAG instructions.

Table 9–17. JTAG Instructions

JTAG Instruction	Cyclone III Device	Cyclone III LS Device
CONFIG_IO	✓	✓
ACTIVE_DISENGAGE	✓	✓
ACTIVE_ENGAGE	✓	✓
EN_ACTIVE_CLK	✓	—
DIS_ACTIVE_CLK	✓	—
APFC_BOOT_ADDR	✓	—
FACTORY *(1)*	—	✓
KEY_PROG_VOL *(2)*	—	✓
KEY_CLR_VREG *(2)*	—	✓

Notes to Table 9–17:

(1) In Cyclone III LS devices, the CONFIG_IO, ACTIVE_DISENGAGE, PULSE_NCONFIG, and PROGRAM instructions are supported, provided that the FACTORY instruction is executed. It is not necessary to execute the FACTORY instruction prior to the JTAG configuration in Cyclone III devices because this instruction is used for Cyclone III LS devices only.

(2) Use the KEY_PROG_VOL and KEY_CLR_VREG instructions for the design security feature. For more information, refer to "Design Security" on page 9–73.

For more information about the JTAG binary instruction code, refer to the *IEEE 1149.1 (JTAG) Boundary-Scan Testing for Cyclone III Devices* chapter.

For Cyclone III LS devices, the device can only allow mandatory JTAG 1149.1 instructions after POR. These instructions are BYPASS, SAMPLE/PRELOAD, EXTEST and FACTORY. To enable the access of other JTAG instructions, issue the FACTORY instruction. The FACTORY instruction puts the device in a state in which it is ready for in-house testing and board-level testing and it must be executed before configuration starts. When this instruction is executed, the CRAM bits content and volatile key are cleared and the device is reset.

I/O Reconfiguration

Use the CONFIG_IO instruction to reconfigure the I/O configuration shift register (IOCSR) chain. This instruction allows you to perform board-level testing prior to configuring the Cyclone III device family or waiting for a configuration device to complete configuration. After the configuration is interrupted and JTAG testing is complete, the part must be reconfigured using the PULSE_NCONFIG JTAG instruction or by pulsing the nCONFIG pin low.

You can issue the CONFIG_IO instruction any time during user mode. The CONFIG_IO instruction cannot be issued when nCONFIG pin is asserted low (during power up) or immediately after issuing a JTAG instruction that triggers reconfiguration. For more information about the wait-time for issuing the CONFIG_IO instruction, refer to Table 9–18.

When using CONFIG_IO instruction, you must meet the following timing restrictions:

- CONFIG_IO instruction cannot be issued during the nCONFIG pin low
- Observe 230 µs minimum wait time after any of the following conditions are met:
  - nCONFIG pin goes high
  - Issuing the PULSE_NCONFIG instruction
  - Issuing the ACTIVE_ENGAGE instruction, before issuing the CONFIG_IO instruction
- Wait 230 µs after power up with nCONFIG pin high before issuing the CONFIG_IO instruction (or wait for the nSTATUS pin to go high)

Table 9–18. Wait Time for Issuing the CONFIG_IO Instruction

Wait Time	Time
Wait time after the nCONFIG pin is released	230 ms
Wait time after PULSE_NCONFIG or ACTIVE_ENGAGE is issued	230 ms

Use the ACTIVE_DISENGAGE instruction with CONFIG_IO instruction to interrupt configuration. Table 9–19 lists the sequence of instructions to use for various CONFIG_IO usage scenarios.

Table 9–19. JTAG CONFIG_IO (without JTAG_PROGRAM) Instruction Flows *(Note 1)* (Part 1 of 2)

JTAG Instruction	Configuration Scheme and Current State of the Cyclone III Device Family											
	Prior to User Mode (Interrupting Configuration)				User Mode				Power Up			
	PS	FPP	AS	AP *(4)*	PS	FPP	AS	AP *(4)*	PS	FPP	AS	AP *(4)*
FACTORY	NA	NA	NA	NA	NA	NA	NA	NA	R	R	R	NA
ACTIVE_DISENGAGE	0	0	0	0	0	0	0	0	—	—	—	—
CONFIG_IO	R	R	R	R	R	R	R	R	NA	NA	NA	NA
JTAG Boundary Scan Instructions (no JTAG_PROGRAM)	0	0	0	0	0	0	0	0	—	—	—	—

Table 9–19. JTAG CONFIG_IO (without JTAG_PROGRAM) Instruction Flows *(Note 1)* (Part 2 of 2)

JTAG Instruction	Configuration Scheme and Current State of the Cyclone III Device Family											
	Prior to User Mode (Interrupting Configuration)				User Mode				Power Up			
	PS	FPP	AS	AP *(4)*	PS	FPP	AS	AP *(4)*	PS	FPP	AS	AP *(4)*
ACTIVE_ENGAGE	A	A	R *(2)*	R *(2)*	A	A	R *(2)*	R *(2)*	—	—	—	—
PULSE_NCONFIG			A *(3)*	A *(3)*			0	0	—	—	—	—
Pulse nCONFIG pin			A *(3)*	A *(3)*			0	0	—	—	—	—
JTAG TAP Reset	R	R	R	R	R	R	R	R	—	—	—	—

Notes to Table 9–19:

(1) "R" indicates that the instruction is to be executed before the next instruction, "O" indicates the optional instruction, "A" indicates that the instruction must be executed, and "NA" indicates that the instruction is not allowed in this mode.

(2) Required if you use ACTIVE_DISENGAGE.

(3) Neither of the instruction is required if you use ACTIVE_ENGAGE.

(4) AP configuration is for Cyclone III devices only.

The CONFIG_IO instruction does not hold the nSTATUS pin low until reconfiguration. You must disengage the active configuration controllers (AS and AP) by issuing the ACTIVE_DISENGAGE and ACTIVE_ENGAGE instructions when the active configuration is interrupted. You must issue the ACTIVE_DISENGAGE instruction alone or prior to the CONFIG_IO instruction if the JTAG_PROGRAM instruction is to be issued later (Table 9–20). This puts the active configuration controllers into the idle state. The active configuration controller is re-engaged after user mode is reached using JTAG programming (Table 9–20).

While executing the CONFIG_IO instruction, all user I/Os are tri-stated.

If reconfiguration after interruption is performed using configuration modes (rather than using JTAG_PROGRAM), it is not necessary to issue the ACTIVE_DISENGAGE instruction prior to CONFIG_IO. You can start reconfiguration by either pulling the nCONFIG pin low for at least 500 ns, or issuing the PULSE_NCONFIG instruction. If the ACTIVE_DISENGAGE instruction was issued and the JTAG_PROGRAM instruction fails to enter user mode, you must issue the ACTIVE_ENGAGE instruction to reactivate the active configuration controller. Issuing the ACTIVE_ENGAGE instruction also triggers the reconfiguration in configuration modes; therefore, it is not necessary to pull the nCONFIG pin low or issue the PULSE_NCONFIG instruction.

ACTIVE_DISENGAGE

The ACTIVE_DISENGAGE instruction places the active configuration controller (AS and AP) into an idle state prior to JTAG programming. The active configuration controller is the AS controller when the MSEL pins are set to AS configuration scheme and the AP controller when the MSEL pins are set to the AP configuration scheme. The two purposes of placing the active controllers in an idle state are:

- To ensure that they are not trying to configure the device in their respective configuration modes during JTAG programming
- To allow the controllers to properly recognize a successful JTAG programming that results in the device reaching user mode

The ACTIVE_DISENGAGE instruction is required before JTAG programming regardless of the current state of the Cyclone III device family if the MSEL pins are set to an active configuration scheme (AS or AP). If the ACTIVE_DISENGAGE instruction is issued during a passive configuration scheme (PS or FPP), it has no effect on the Cyclone III device family. Similarly, the CONFIG_IO instruction is issued after an ACTIVE_DISENGAGE instruction, but is no longer required to properly halt configuration. Table 9–20 lists the required, recommended, and optional instructions for each configuration mode. The ordering of the required instructions is a hard requirement and must be met to ensure functionality.

Table 9–20. JTAG Programming Instruction Flows *(Note 1)*

JTAG Instruction	Configuration Scheme and Current State of the Cyclone III Device											
	Prior to User Mode (Interrupting Configuration)				User Mode				Power Up			
	PS	FPP	AS	AP *(2)*	PS	FPP	AS	AP *(2)*	PS	FPP	AS	AP *(2)*
FACTORY	NA	NA	NA	NA	NA	NA	NA	NA	R	R	R	NA
ACTIVE_DISENGAGE	O	O	R	R	O	O	O	R	O	O	R	R
CONFIG_IO	Rc	Rc	O	O	O	O	O	O	NA	NA	NA	NA
Other JTAG instructions	O	O	O	O	O	O	O	O	O	O	O	O
JTAG_PROGRAM	R	R	R	R	R	R	R	R	R	R	R	R
CHECK_STATUS	Rc	Rc	Rc	Rc	Rc	Rc	Rc	Rc	Rc	Rc	Rc	Rc
JTAG_STARTUP	R	R	R	R	R	R	R	R	R	R	R	R
JTAG TAP Reset/ other instruction	R	R	R	R	R	R	R	R	R	R	R	R

Notes to Table 9–20:

(1) "R" indicates that the instruction is required to be executed before the next instruction, "O" indicates the optional instruction, "Rc" indicates the recommended instruction, and "NA" indicates that the instruction is not allowed to be executed in this mode.

(2) AP configuration is for Cyclone III devices only.

In AS or AP configuration schemes, the ACTIVE_DISENGAGE instruction puts the active configuration controllers into idle state. If a successful JTAG programming is executed, the active controllers are automatically re-engaged after user mode is reached using JTAG programming. This causes the active controllers to transition to their respective user mode states.

If JTAG programming fails to get the Cyclone III device family to enter user mode and re-engage active programming, there are available methods to achieve this for the AS or AP configuration schemes:

- When in the AS configuration scheme, you can re-engage the AS controller by moving the JTAG TAP controller to the reset state or by issuing the ACTIVE_ENGAGE instruction.
- When in the AP configuration scheme, the only way to re-engage the AP controller is to issue the ACTIVE_ENGAGE instruction. In this case, asserting the nCONFIG pin does not re-engage either active controller.

ACTIVE_ENGAGE

The ACTIVE_ENGAGE instruction allows you to re-engage a disengaged active controller. You can issue this instruction any time during configuration or user mode to re-engage an already disengaged active controller as well as trigger reconfiguration of the Cyclone III device family in the active configuration scheme specified by the MSEL pin settings.

The ACTIVE_ENGAGE instruction functions as the PULSE_NCONFIG instruction when the device is in passive configuration schemes (PS or FPP). The nCONFIG pin is disabled when the ACTIVE_ENGAGE instruction is issued.

Altera does not recommend using the ACTIVE_ENGAGE instruction but it is provided as a fail-safe instruction for re-engaging the active configuration (AS or AP) controllers.

Overriding the Internal Oscillator

This feature is used for Cyclone III devices only and allows you to override the internal oscillator during the active configuration scheme. The active configuration (AS and AP) controllers use the internal oscillator as the clock source. You can change the clock source to CLKUSR through JTAG instruction.

The EN_ACTIVE_CLK and DIS_ACTIVE_CLK JTAG instructions toggle on or off whether the active clock is sourced from the CLKUSR pin or the internal configuration oscillator. To source the active clock from the CLKUSR pin, issue the EN_ACTIVE_CLK instruction. This causes the CLKUSR pin to become the active clock source. When using the EN_ACTIVE_CLK instruction, the internal oscillator must be enabled for the clock change to occur. By default, the configuration oscillator is disabled after configuration and initialization is complete and the device has entered user mode. However, the internal oscillator is enabled in user mode by either one of the following conditions:

- A reconfiguration event (for example, driving nCONFIG low)
- Remote update is enabled
- Error detection is enabled

You must clock the CLKUSR pin at two times the expected DCLK frequency. The CLKUSR pin allows a maximum frequency of 80 MHz (40 MHz DCLK). Normally, a test instrument uses the CLKUSR pin when it wants to drive its own clock to control the AS state machine.

To revert the clock source back to the configuration oscillator, issue the DIS_ACTIVE_CLK instruction. After you issue the DIS_ACTIVE_CLK instruction, you must continue to clock the CLKUSR pin for 10 clock cycles. Otherwise, even toggling the nCONFIG pin does not revert the clock source and reconfiguration does not occur. A POR reverts the clock source back to the configuration oscillator. Toggling the nCONFIG pin or driving the JTAG state machine to the reset state does not revert the clock source.

EN_ACTIVE_CLK

The EN_ACTIVE_CLK instruction causes the CLKUSR pin signal to replace the internal oscillator as the clock source. When using the EN_ACTIVE_CLK instruction, the internal oscillator must be enabled for the clock change to occur. After this instruction is issued, other JTAG instructions can be issued while the CLKUSR pin signal remains as the clock source. The clock source is only reverted back to the internal oscillator by issuing the DIS_ACTIVE_CLK instruction or a POR.

DIS_ACTIVE_CLK

The DIS_ACTIVE_CLK instruction breaks the CLKUSR enable latch set by the EN_ACTIVE_CLK instruction and causes the clock source to revert back to the internal oscillator. After the DIS_ACTIVE_CLK instruction is issued, you must continue to clock the CLKUSR pin for 10 clock cycles.

The CLKUSR pin must be clocked at two times the expected DCLK frequency. The CLKUSR pin allows a maximum frequency of 80 MHz (40 MHz DCLK).

Changing the Start Boot Address of the AP Flash

In the AP configuration scheme, for Cyclone III devices only, you can change the default configuration boot address of the parallel flash memory to any desired address using the APFC_BOOT_ADDR JTAG instruction.

APFC_BOOT_ADDR

The APFC_BOOT_ADDR instruction is for Cyclone III devices only and allows you to define a start boot address for the parallel flash memory in the AP configuration scheme.

This instruction shifts in a start boot address for the AP flash. When this instruction becomes the active instruction, the TDI and TDO pins are connected through a 22-bit active boot address shift register. The shifted-in boot address bits get loaded into the 22-bit AP boot address update register, which feeds into the AP controller. The content of the AP boot address update register can be captured and shifted-out of the active boot address shift register from TDO.

The boot address in the boot address shift register and update register are shifted to the right (in the LSB direction) by two bits versus the intended boot address. The reason for this is that the two LSB of the address are not accessible. When this boot address is fed into the AP controller, two 0s are attached in the end as LSB, thereby pushing the shifted-in boot address to the left by two bits, which become the actual AP boot address the AP controller gets.

If you have enabled the remote update feature, the APFC_BOOT_ADDR instruction sets the boot address for the factory configuration only.

☞ The APFC_BOOT_ADDR instruction is retained after reconfiguration while the system board is still powered on. However, you must reprogram the instruction whenever you restart the system board.

Device Configuration Pins

Table 9–21 through Table 9–23 describe the connections and functionality of all the configuration-related pins on Cyclone III device family.

Table 9–21 lists the Cyclone III device family pin configuration.

Table 9–21. Cyclone III Device Family Configuration Pin Summary (Part 1 of 2)

Bank	Description	Input/Output	Dedicated	Powered By	Configuration Mode
1	FLASH_nCE, nCSO	Output	—	V_{CCIO}	AS, AP *(2)*
6	CRC_ERROR	Output	—	V_{CCIO}/Pull-up *(1)*	Optional, all modes
1	DATA[0]	Input	Yes	V_{CCIO}	PS, FPP, AS
		Bidirectional		V_{CCIO}	AP *(2)*
1	DATA[1], ASDO	Input	—	V_{CCIO}	FPP
		Output		V_{CCIO}	AS
		Bidirectional		V_{CCIO}	AP *(2)*
8	DATA[7..2]	Input	—	V_{CCIO}	FPP
		Bidirectional		V_{CCIO}	AP *(2)*
8	DATA[15..8]	Bidirectional		V_{CCIO}	AP *(2)*
6	INIT_DONE	Output	—	Pull-up	Optional, all modes
1	nSTATUS	Bidirectional	Yes	Pull-up	All modes
1	nCE	Input	Yes	V_{CCIO}	All modes
1	DCLK	Input	Yes	V_{CCIO}	PS, FPP
		Output		V_{CCIO}	AS, AP *(2)*
6	CONF_DONE	Bidirectional	Yes	Pull-up	All modes
1	TDI	Input	Yes	V_{CCIO}	JTAG
1	TMS	Input	Yes	V_{CCIO}	JTAG
1	TCK	Input	Yes	V_{CCIO}	JTAG
1	nCONFIG	Input	Yes	V_{CCIO}	All modes
6	CLKUSR	Input	—	V_{CCIO}	Optional
6	nCEO	Output	—	V_{CCIO}	Optional, all modes
6	MSEL[3..0]	Input	Yes	V_{CCINT}	All modes
1	TDO	Output	Yes	V_{CCIO}	JTAG
7	PADD[14..0]	Output	—	V_{CCIO}	AP *(2)*
8	PADD[19..15]	Output	—	V_{CCIO}	AP *(2)*
6	PADD[23..20]	Output	—	V_{CCIO}	AP *(2)*
1	nRESET	Output	—	V_{CCIO}	AP *(2)*
6	nAVD	Output	—	V_{CCIO}	AP *(2)*
6	nOE	Output	—	V_{CCIO}	AP *(2)*
6	nWE	Output	—	V_{CCIO}	AP *(2)*

Table 9–21. Cyclone III Device Family Configuration Pin Summary (Part 2 of 2)

Bank	Description	Input/Output	Dedicated	Powered By	Configuration Mode
5	DEV_OE	Input	—	V_{CCIO}	Optional, AP *(2)*
5	DEV_CLRn	Input	—	V_{CCIO}	Optional, AP *(2)*

Notes to Table 9–21:

(1) In Cyclone III devices, the CRC_ERROR pin is a dedicated output by default. Optionally, you can enable the CRC_ERROR pin as an open-drain output in the **CRC Error Detection** tab from the **Device and Pin Options** dialog box.

(2) AP configuration is for Cyclone III devices only.

Table 9–22 lists the dedicated configuration pins that must be connected properly on your board for successful configuration. Some of these pins may not be required for your configuration scheme.

Table 9–22. Dedicated Configuration Pins on Cyclone III Device Family (Part 1 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
MSEL [3..0]	N/A	All	Input	4-bit configuration input that sets the Cyclone III device family configuration scheme. These pins must be hardwired to V_{CCA} or GND. The MSEL[3..0] pins have internal 9-kΩ pull-down resistors that are always active. Some of the smaller devices or package options of Cyclone III devices do not have the MSEL[3] pin; therefore, the AP configuration scheme is not supported.
nCONFIG	N/A	All	Input	Configuration control input. Pulling this pin low with external circuitry during user mode causes the Cyclone III device family to lose its configuration data, enter a reset state, and tri-state all I/O pins. Returning this pin to a logic-high level starts a reconfiguration.
nSTATUS	N/A	All	Bidirectional open-drain	The Cyclone III device family drives nSTATUS low immediately after power-up and releases it after the POR time. ■ Status output. If an error occurs during configuration, nSTATUS is pulled low by the target device. ■ Status input. If an external source (for example, another Cyclone III device family) drives the nSTATUS pin low during configuration or initialization, the target device enters an error state. Driving nSTATUS low after configuration and initialization does not affect the configured device. If you use a configuration device, driving nSTATUS low causes the configuration device to attempt to configure the device, but because the device ignores transitions on nSTATUS in user mode, the device does not reconfigure. To start a reconfiguration, nCONFIG must be pulled low.

Table 9–22. Dedicated Configuration Pins on Cyclone III Device Family (Part 2 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
CONF_DONE	N/A	All	Bidirectional open-drain	■ Status output. The target Cyclone III device family drives the CONF_DONE pin low before and during configuration. After all configuration data is received without error and the initialization cycle starts, the target device releases CONF_DONE. ■ Status input. After all data is received and CONF_DONE goes high, the target device initializes and enters user mode. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize. Driving CONF_DONE low after configuration and initialization does not affect the configured device. Do not connect bus holds or ADC to the CONF_DONE pin.
nCE	N/A	All	Input	Active-low chip enable. The nCE pin activates the Cyclone III device family with a low signal to allow configuration. The nCE pin must be held low during configuration, initialization, and user-mode. In a single-device configuration, it must be tied low. In a multi-device configuration, nCE of the first device is tied low while its nCEO pin is connected to the nCE pin of the next device in the chain. The nCE pin must also be held low for successful JTAG programming of the device.
nCEO	N/A if option is on. I/O if option is off.	All	Output open drain	Output that drives low when configuration is complete. In a single-device configuration, you can leave this pin floating or use it as a user I/O pin after configuration. In a multi-device configuration, this pin feeds the nCE pin of the next device. The nCEO of the last device in the chain is left floating or is used as a user I/O pin after configuration. If you use the nCEO pin to feed the nCE pin of the next device, use an external 10-kΩ pull-up resistor to pull the nCEO pin high to the V_{CCIO} voltage of its I/O bank to help the internal weak pull-up resistor. If you use the nCEO pin as a user I/O pin after configuration, set the state of the pin on the **Dual-Purpose Pin** settings.
FLASH_nCE, nCSO	I/O	AS, AP *(1)*	Output	Output control signal from the Cyclone III device family to the serial configuration device in AS mode that enables the configuration device. This pin functions as the nCSO pin in AS mode and the FLASH_NCE pin in AP mode. Output control signal from the Cyclone III device to the parallel flash in AP mode that enables the flash. Connects to the CE# pin on the Numonyx P30 or P33 flash. *(1)* This pin has an internal pull-up resistor that is always active.

Table 9–22. Dedicated Configuration Pins on Cyclone III Device Family (Part 3 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
DCLK	N/A	PS, FPP, AS, AP *(1)*	Input (PS, FPP). Output (AS, AP*(1)*)	In PS and FPP configuration, DCLK is the clock input used to clock data from an external source into the target Cyclone III device family. Data is latched into the device on the rising edge of DCLK. In AS mode, DCLK is an output from the Cyclone III device family that provides timing for the configuration interface, it has an internal pull-up resistor (typically 25 kΩ) that is always active. In AP mode, DCLK is an output from the Cyclone III device that provides timing for the configuration interface. *(1)* In active configuration schemes (AS or AP), this pin will be driven into an inactive state after configuration completes. Alternatively, in active schemes, you can use this pin as a user I/O during user mode. In passive schemes (PS or FPP) that use a control host, DCLK must be driven either high or low, whichever is more convenient. In passive schemes, you cannot use DCLK as a user I/O in user mode. Toggling this pin after configuration does not affect the configured device
DATA[0]	I/O	PS, FPP, AS, AP *(1)*	Input (PS, FPP, AS). Bidirectional (AP)*(1)*	Data input. In serial configuration modes, bit-wide configuration data is presented to the target Cyclone III device family on the DATA[0] pin. In AS mode, DATA[0] has an internal pull-up resistor that is always active. After AS configuration, DATA[0] is a dedicated input pin with optional user control. After PS or FPP configuration, DATA[0] is available as a user I/O pin and the state of this pin depends on the **Dual-Purpose Pin** settings. After AP configuration, DATA[0] is a dedicated bidirectional pin with optional user control. *(1)*
DATA[1], ASDO	I/O	FPP, AS, AP *(1)*	Input (FPP), Output (AS). Bidirectional (AP) *(1)*	Data input in non-AS mode. Control signal from the Cyclone III device family to the serial configuration device in AS mode used to read out configuration data. The DATA[1] pin functions as the ASDO pin in AS mode. In AS mode, DATA[1] has an internal pull-up resistor that is always active. After AS configuration, DATA[1] is a dedicated output pin with optional user control. In PS configuration scheme, DATA[1] functions as user I/O pin during configuration, which means it is tri-stated. After FPP configuration, DATA[1] is available as a user I/O pin and the state of this pin depends on the **Dual-Purpose Pin** settings. In AP configuration scheme, which is for Cyclone III devices only, the byte-wide or word-wide configuration data is presented to the target Cyclone III device on DATA[7..0] or DATA[15..0], respectively. After AP configuration, DATA[1] is a dedicated bidirectional pin with optional user control. *(1)*

Table 9–22. Dedicated Configuration Pins on Cyclone III Device Family (Part 4 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
DATA[7..2]	I/O	FPP, AP *(1)*	Inputs (FPP). Bidirectional (AP) *(1)*	Data inputs. In AS or PS configuration schemes, they function as user I/O pins during configuration, which means they are tri-stated. After FPP configuration, DATA[7..2] are available as user I/O pins and the state of these pin depends on the **Dual-Purpose Pin** settings. The byte-wide or word-wide configuration data is presented to the target Cyclone III device on DATA[7..0] or DATA[15..0], respectively, in the AP configuration scheme (for Cyclone III devices only). After AP configuration, DATA[7..2] are dedicated bidirectional pins with optional user control. *(1)*
DATA[15..8]	I/O	AP *(1)*	Bidirectional	Data inputs. Word-wide configuration data is presented to the target Cyclone III device on DATA[15..0]. In PS, FPP, or AS configuration schemes, they function as user I/O pins during configuration, which means they are tri-stated. After AP configuration, DATA[15:8] are dedicated bidirectional pins with optional user control.
PADD[23..0]	I/O	AP *(1)*	Output	24-bit address bus from the Cyclone III device to the parallel flash in AP mode. Connects to the A[24:1] bus on the Numonyx P30 or P33 flash.
nRESET	I/O	AP *(1)*	Output	Active-low reset output. Driving the nRESET pin low resets the parallel flash. Connects to the RST# pin on the Numonyx P30 or P33 flash.
nAVD	I/O	AP *(1)*	Output	Active-low address valid output. Driving the nAVD pin low during a read or write operation indicates to the parallel flash that valid address is present on the PADD[23..0] address bus. Connects to the ADV# pin on the Numonyx P30 or P33 flash.
nOE	I/O	AP *(1)*	Output	Active-low output enable to the parallel flash. Driving the nOE pin low during a read operation enables the parallel flash outputs (DATA[15..0]). Connects to the OE# pin on the Numonyx P30 or P33 flash.
nWE	I/O	AP *(1)*	Output	Active-low write enable to the parallel flash. Driving the nWE pin low during a write operation indicates to the parallel flash that data on the DATA[15..0] bus is valid. Connects to the WE# pin on the Numonyx P30 or P33 flash.

Note to Table 9–22:

(1) AP configuration scheme is for Cyclone III devices only.

Table 9–23 lists the optional configuration pins. If these optional configuration pins are not enabled in the Quartus II software, they are available as general-purpose user I/O pins. Therefore, during configuration, these pins function as user I/O pins and are tri-stated with weak pull-up resistors.

Table 9–23. Optional Configuration Pins

Pin Name	User Mode	Pin Type	Description
CLKUSR	N/A if option is on. I/O if option is off.	Input	Optional user-supplied clock input synchronizes the initialization of one or more devices. This pin is enabled by turning on the **Enable user-supplied start-up clock** (**CLKUSR**) option in the Quartus II software.
INIT_DONE	N/A if option is on. I/O if option is off.	Output open-drain	Status pin used to indicate when the device has initialized and is in user-mode. When nCONFIG is low and during the beginning of configuration, the INIT_DONE pin is tri-stated and pulled high due to an external 10-kΩ pull-up resistor. After the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high and the device enters user mode. Thus, the monitoring circuitry must be able to detect a low-to-high transition. This pin is enabled by turning on the **Enable INIT_DONE output** option in the Quartus II software. The functionality of this pin changes if the **Enable OCT_DONE** option is enabled in the Quartus II software. This option controls whether the INIT_DONE signal is gated by the OCT_DONE signal, which indicates the Power-Up OCT calibration is complete. If this option is turned off, the INIT_DONE signal is not gated by the OCT_DONE signal
DEV_OE	N/A if option is on. I/O if option is off.	Input	Optional pin that allows you to override all tri-states on the device. When this pin is driven low, all I/O pins are tri-stated; when this pin is driven high, all I/O pins behave as programmed. This pin is enabled by turning on the **Enable device-wide output enable (DEV_OE)** option in the Quartus II software.
DEV_CLRn	N/A if option is on. I/O if option is off.	Input	Optional pin that allows you to override all clears on all device registers. When this pin is driven low, all registers are cleared; when this pin is driven high, all registers behave as programmed. This pin is enabled by turning on the **Enable device-wide reset (DEV_CLRn)** option in the Quartus II software.

Design Security

The design security feature is for Cyclone III LS devices only. The design security feature is not supported in Cyclone III devices.

Cyclone III LS Design Security Protection

Cyclone III LS device designs are protected from copying, reverse engineering, and tampering using configuration bitstream encryption and anti-tamper features.

Security Against Copying

The volatile key is securely stored in the Cyclone III LS device and cannot be read out through any interfaces. The information of your design cannot be copied because the configuration file read-back feature is not supported in Cyclone III LS devices.

Security Against Reverse Engineering

Reverse engineering from an encrypted configuration file is very difficult and time consuming because Cyclone III LS configuration file formats are proprietary and the file contains million of bits which require specific decryption. Reverse engineering the Cyclone III LS device is just as difficult because the device is manufactured on the advanced 60-nm process technology.

Security Against Tampering

Cyclone III LS devices support the following anti-tamper features:

- Ability to limit JTAG instruction set and provides protection against configuration data readback over the JTAG port
- Ability to clear contents of FPGA logic, configuration memory, user memory, and volatile key
- Error detection (ED) cycle indicator to core Cyclone III LS devices provide a pass or fail indicator at every ED cycle and visibility over intentional or unintentional change of CRAM bits.

For more information about anti-tamper protection for Cyclone III LS devices, refer to *AN 593: Anti-Tamper Protection for Cyclone III LS Devices*.

For more information about the implementation of secure configuration flow in Quartus II, refer to *AN 589: Using Design Security Feature in Cyclone III LS Devices*.

AES Decryption Block

The main purpose of the AES decryption block is to decrypt the configuration bitstream prior to entering configuration. Prior to receiving encrypted data, you must enter and store the 256-bit volatile key in the device with battery backup. The key is scrambled prior to storing it in the key storage to make it more difficult for anyone to retrieve the stored key using de-capsulation of the device.

Key Storage

Cyclone III LS devices support volatile key programming. Table 9–24 lists the volatile key features.

Table 9–24. Security Key Features

Volatile Key Features	Description
Key programmability	Reprogrammable and erasable
External battery	Required
Key programming method *(1)*	On-board
Design protection	Secure against copying, reverse engineering, and tampering

Note to Table 9–24:

(1) Key programming is carried out using the JTAG interface.

AES volatile key zeroization is supported in Cyclone III LS devices. The volatile key clear and key program JTAG instructions from the device core is supported to protect Cyclone III LS devices against tampering. You can clear and reprogram the key from the device core whenever tampering attempt is detected by executing the `KEY_CLR_VREG` and `KEY_PROG_VOL` JTAG instructions to clear and reprogram the volatile key, and then reset the Cyclone III LS device by pulling the `nCONFIG` pin low for at least 500 ns. When `nCONFIG` returns to a logic-high level and `nSTATUS` is released by the Cyclone III LS device, reconfiguration begins to configure the Cyclone III LS device with a benign or unencrypted configuration file. After configuration is successfully completed, observe the `cyclecomplete` signal from error detection block to ensure that reconfigured CRAM bits content is correct for at least one error detection cycle. You can also observe the `cyclecomplete` and `crcerror` signals for any unintentional CRAM bits change.

`cyclecomplete` is a signal that is routed from the error detection block to the core for the purpose of every complete error detection cycle. You must include the `cycloneiiils_crcblock` WYSIWYG atom in your design to use the `cyclecomplete` signal. For more information about the SEU mitigation, refer to the *SEU Mitigation in Cyclone III Devices* chapter.

V_{CCBAT} is a dedicated power supply for the volatile key storage and not shared with other on-chip power supplies, such as V_{CCIO} or V_{CC}. V_{CCBAT} continuously supplies power to the volatile register regardless of the on-chip supply condition. The nominal voltage for this supply is 3.0 V, while its valid operating range is from 1.2 to 3.3 V. If you do not use the volatile security key, you may connect the V_{CCBAT} to a 1.8-V, 2.5-V, or 3.0-V power supply.

After power-up, wait for 200 ms (Standard POR) or 9 ms (Fast POR) before beginning the key programming to ensure that V_{CCBAT} is at its full rail.

As an example, BR1220 (-30°C to +80°C) and BR2477A (-40 C to +125°C) are lithium coin-cell type batteries used for volatile key storage purposes.

For more information about the battery specifications, refer to the *Cyclone III LS Device Data Sheet* chapter.

Cyclone III LS Design Security Solution

Cyclone III LS devices are SRAM-based devices. To provide design security, Cyclone III LS devices require a 256-bit volatile key for configuration bitstream encryption.

The Cyclone III LS design security feature provides routing architecture optimization for design separation flow with the Quartus II software. Design separation flow achieves both physical and functional isolation between design partitions.

For more information about the design separation flow, refer to *AN 567: Quartus II Design Separation Flow.*

You can carry out secure configuration in Steps 1–3, as shown in Figure 9–31:

1. Generate the encryption key programming file and encrypt the configuration data.

   The Quartus II configuration software uses the user-defined 256-bit volatile keys to generate a key programming file and an encrypted configuration file. The encrypted configuration file is stored in an external memory, such as a flash memory or a configuration device.

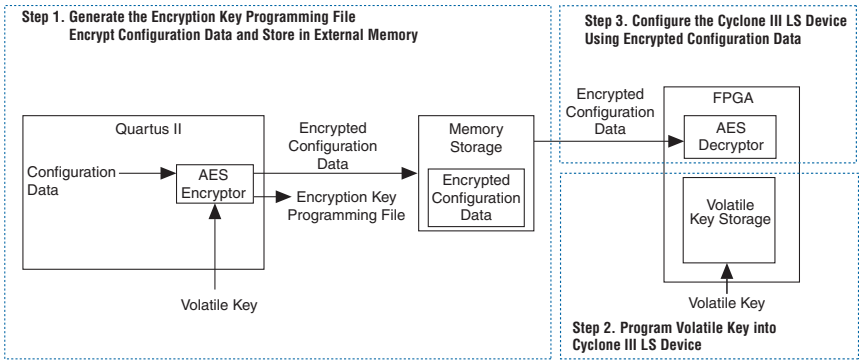
2. Program the volatile key into the Cyclone III LS device.

   Program the user-defined 256-bit volatile keys into the Cyclone III LS device through the JTAG interface.

3. Configure the Cyclone III LS device.

   At system power-up, the external memory device sends the encrypted configuration file to the Cyclone III LS device.

Figure 9–31. Cyclone III LS Secure Configuration Flow *(Note 1)*



Note to Figure 9–31:

(1) Step 1, Step 2, and Step 3 correspond to the procedure detailed in "Cyclone III LS Design Security Solution".

Available Security Modes

There are several security modes available on Cyclone III LS devices, they are:

- Volatile Key
- No Key Operation
- FACTORY Mode

Volatile Key

Secure operation with volatile key programmed and required external battery—this mode accepts both encrypted and unencrypted configuration bitstreams. Use the unencrypted configuration bitstream support for board-level testing only.

No Key Operation

Only unencrypted configuration bitstreams are allowed to configure the device.

FACTORY Mode

After power up, Cyclone III LS devices must be in FACTORY mode to program the volatile key. The FACTORY private JTAG instruction must be issued after the device successfully exits from POR and before the device starts loading the core configuration data to enable access to all other instructions from the JTAG pins. The device configuration data and AES volatile key are cleared if the FACTORY instruction is executed.

Table 9–25 lists the configuration bitstream and the configuration mode supported for each security mode.

Table 9–25. Security Modes Supported

Mode	Function	Configuration File	Allowed Configuration Mode
Volatile Key	Secure	Encrypted	PS with AES (without decompression). FPP with AES (without decompression). Remote update fast AS with AES (without decompression). Fast AS (without decompression).
	Board-Level Testing	Unencrypted	All configuration modes that do not engage the design security feature.
No Key	—	Unencrypted	All configuration modes that do not engage the design security feature.
FACTORY	Volatile Key Programming	—	—

Remote System Upgrade

Cyclone III devices support remote system upgrade in AS and AP configuration schemes. Cyclone III LS devices support remote system upgrade in the AS configuration scheme only. Remote system upgrade can also be implemented with advanced Cyclone III features such as real-time decompression of configuration data in the AS configuration scheme.

- The serial configuration device uses the AS configuration scheme to configure Cyclone III or Cyclone III LS devices
- The supported parallel flash uses the AP configuration scheme to configure Cyclone III devices
- Remote system upgrade is not supported in the multi-device configuration chain for any configuration scheme.

Functional Description

The dedicated remote system upgrade circuitry in Cyclone III device family manages remote configuration and provides error detection, recovery, and status information. User logic or a Nios® II processor implemented in the Cyclone III device family logic array provides access to the remote configuration data source and an interface to the configuration memory.

Configuration memory refers to serial configuration devices (EPCS) or supported parallel flash memory, and depends on the configuration scheme that you use.

The remote system upgrade process of Cyclone III device family involves the following steps:

1. A Nios II processor (or user logic) implemented in the Cyclone III device family logic array receives new configuration data from a remote location. The connection to the remote source is a communication protocol such as the transmission control protocol/Internet protocol (TCP/IP), peripheral component interconnect (PCI), user datagram protocol (UDP), universal asynchronous receiver/transmitter (UART), or a proprietary interface.
2. The Nios II processor (or user logic) writes this new configuration data into a configuration memory.
3. The Nios II processor (or user logic) starts a reconfiguration cycle with the new or updated configuration data.
4. The dedicated remote system upgrade circuitry detects and recovers from any error that might occur during or after the reconfiguration cycle, and provides error status information to the user design.

Figure 9–32 shows the steps required for performing remote configuration updates (the numbers in Figure 9–32 coincide with steps 1–4).

Figure 9–32. Functional Diagram of Cyclone III Device Family Remote System Upgrade

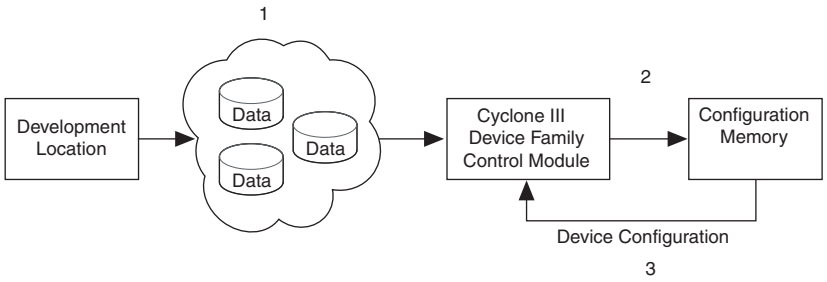
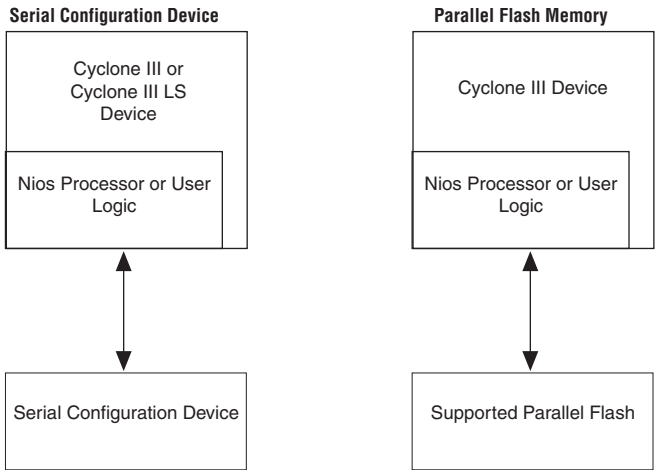


Figure 9–33 shows the block diagrams to implement remote system upgrade with the AS and AP configuration schemes.

Figure 9–33. Remote System Upgrade Block Diagrams for AS and AP Configuration Schemes


Serial Configuration Device
Cyclone III or Cyclone III LS Device
Nios Processor or User Logic
Serial Configuration Device
Parallel Flash Memory
Cyclone III Device
Nios Processor or User Logic
Supported Parallel Flash

Remote system upgrade only supports single-device configuration.

When using remote system upgrade in Cyclone III devices, you must set the mode select pins (`MSEL [3.0]`) to the AS or AP configuration scheme. When using remote system upgrade in Cyclone III LS devices, you must set `MSEL [3..0]` to the AS configuration scheme. The MSEL pin setting in remote system upgrade mode is the same as standard configuration mode. Standard configuration mode refers to normal Cyclone III device family configuration mode with no support for remote system upgrades, and the remote system upgrade circuitry is disabled. When using remote system upgrade in Cyclone III device family, you must enable the **remote update mode** option setting in the Quartus II software. For more information, refer to "Enabling Remote Update" on page 9–79.

Enabling Remote Update

You can enable or disable remote update for Cyclone III device family in the Quartus II software before design compilation (in the Compiler Settings menu). To enable remote update in the compiler settings of the project, perform the following steps in the Quartus II software:

1. On the Assignments menu, click **Device**. The **Settings** dialog box appears.
2. Click **Device and Pin Options**. The **Device and Pin Options** dialog box appears.
3. Click the **Configuration** tab.
4. From the **Configuration Mode** list, select **Remote**.
5. Click **OK**.
6. In the **Settings** dialog box, click **OK**.

Configuration Image Types

When using remote system upgrade, Cyclone III device family configuration bitstreams are classified as factory configuration images or application configuration images. An image, also referred to as a configuration, is a design loaded into the device that performs certain user-defined functions. Each device in your system requires one factory image with the addition of one or more application images. The factory image is a user-defined fall-back, or safe, configuration and is responsible for administering remote updates with dedicated circuitry. Application images implement user-defined functionality in the target Cyclone III device family. You can include the default application image functionality in the factory image.

Remote System Upgrade Mode

Overview

In remote update mode, the Cyclone III device family loads the factory configuration image after power-up. The user-defined factory configuration determines which application configuration is to be loaded and triggers a reconfiguration cycle. The factory configuration can also contain application logic.

When used with configuration memory, remote update mode allows an application configuration to start at any flash sector boundary. Additionally, the remote update mode features a user watchdog timer that can detect functional errors in an application configuration.

Remote Update Mode

When a Cyclone III device family is first powered up in remote update in the AS configuration scheme, it loads the factory configuration located at address `boot_address[23:0] = 24b'0`. Altera recommends storing the factory configuration image for your system at boot address `24b'0` when using the AS configuration scheme. A factory configuration image is a bitstream for Cyclone III device family in your system that is programmed during production and is the fall-back image when an error occurs. This image is stored in non-volatile memory and is never updated or modified using remote access. This corresponds to the start address location 0x000000 in the serial configuration device.

When you use the AP configuration in Cyclone III devices, the Cyclone III device loads the default factory configuration located at the following address after device power-up in remote update mode:

boot_address[23:0] = 24'h010000 = 24'b1 0000 0000 0000 0000

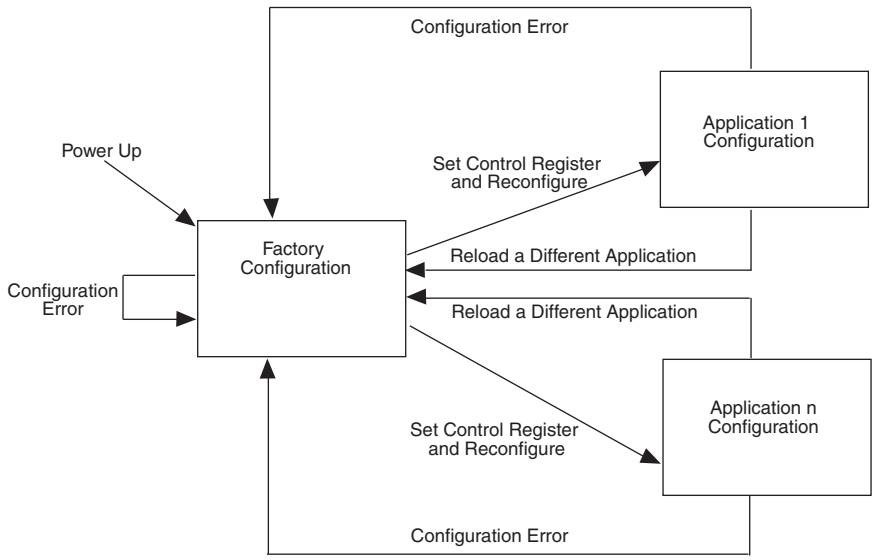
You can change the default factory configuration address to any desired address using the APFC_BOOT_ADDR JTAG instruction. The factory configuration image is stored in non-volatile memory and is never updated or modified using remote access. This corresponds to the default start address location 0x010000 represented in 16-bit word addressing (or the updated address if the default address is changed) in the supported parallel flash memory. For more information about the application of the APFC_BOOT_ADDR JTAG instruction in AP configuration scheme, refer to "JTAG Instructions" on page 9–61.

The factory configuration image is user designed and contains soft logic (Nios II processor or state machine and the remote communication interface) to:

- Process any errors based on status information from the dedicated remote system upgrade circuitry
- Communicate with the remote host and receive new application configurations and store the new configuration data in the local non-volatile memory device
- Determine which application configuration is to be loaded into the Cyclone III device family
- Enable or disable the user watchdog timer and load its time-out value (optional)
- Instruct the dedicated remote system upgrade circuitry to start a reconfiguration cycle

Figure 9–34 shows the transitions between the factory and application configurations in remote update mode.

Figure 9–34. Transitions Between Configurations in Remote Update Mode



After power up or a configuration error, the factory configuration logic writes the remote system upgrade control register to specify the address of the application configuration to be loaded. The factory configuration also specifies whether or not to enable the user watchdog timer for the application configuration and, if enabled, specifies the timer setting.

Only valid application configurations designed for remote update mode include the logic to reset the timer in user mode. For more information about the user watchdog timer, refer to "User Watchdog Timer" on page 9–87.

If there is an error while loading the application configuration, the remote system upgrade status register is written by the dedicated remote system upgrade circuitry of the Cyclone III device family, specifying the cause of the reconfiguration.

The following actions cause the remote system upgrade status register to be written:

- `nSTATUS` driven low externally
- Internal CRC error
- User watchdog timer time-out
- A configuration reset (logic array `nCONFIG` signal or external `nCONFIG` pin assertion)

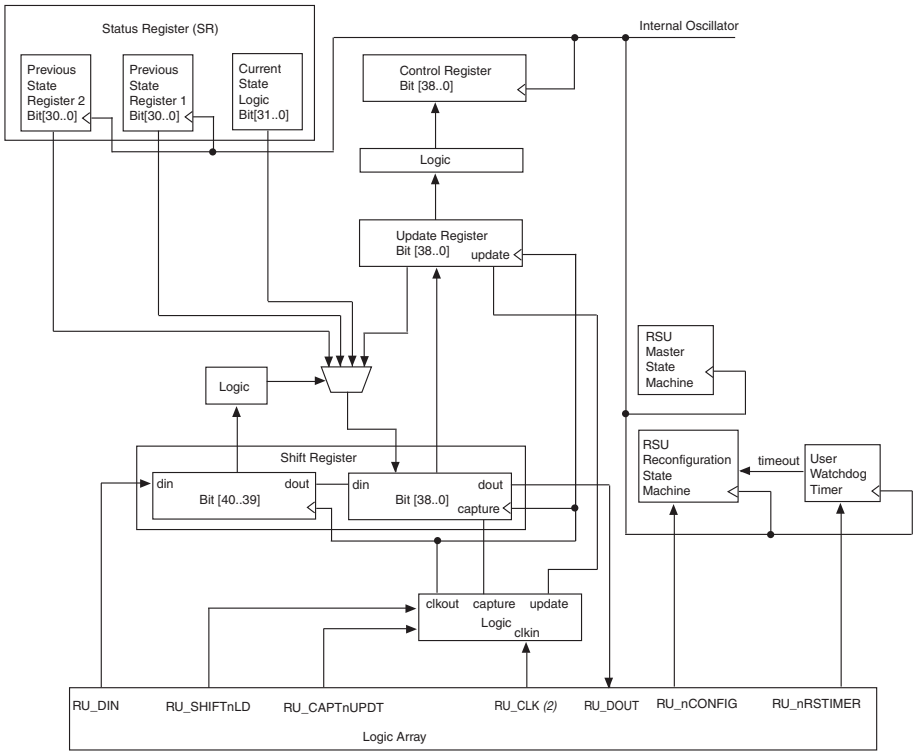
Cyclone III device family automatically load the factory configuration located at address `boot_address[23:0] = 24'b0` for the AS configuration scheme, and default address boot_address[23:0] = 24'h010000 (or the updated address if the default address is changed) for the AP configuration scheme. This user-designed factory configuration reads the remote system upgrade status register to determine the reason for reconfiguration. Then the factory configuration takes the appropriate error recovery steps and writes to the remote system upgrade control register to determine the next application configuration to be loaded.

When Cyclone III device family successfully load the application configuration, the devices enter user mode. In user mode, the soft logic (Nios II processor or state machine and the remote communication interface) assists the Cyclone III device family in determining when a remote system update is arriving. When a remote system update arrives, the soft logic receives the incoming data, writes it to the configuration memory device, and triggers the device to load the factory configuration. The factory configuration reads the remote system upgrade status register, determines the valid application configuration to load, writes the remote system upgrade control register accordingly, and starts system reconfiguration.

Dedicated Remote System Upgrade Circuitry

This section explains the implementation of the Cyclone III device family remote system upgrade dedicated circuitry. The remote system upgrade circuitry is implemented in hard logic. This dedicated circuitry interfaces to the user-defined factory application configurations implemented in the Cyclone III device family logic array to provide the complete remote configuration solution. The remote system upgrade circuitry contains the remote system upgrade registers, a watchdog timer, and state machines that control those components. Figure 9–35 shows the data path of the remote system upgrade block.

Figure 9–35. Remote System Upgrade Circuit Data Path *(Note 1)*



Notes to Figure 9–35:

(1) RU_DOUT, RU_SHIFTnLD, RU_CAPTnUPDT, RU_CLK, RU_DIN, RU_nCONFIG, and RU_nRSTIMER signals are internally controlled by the ALTREMOTE_UPDATE megafunction.

(2) RU_CLK refers to ALTREMOTE_UPDATE megafunction block "clock" input. For more information, refer to the *Remote Update Circuitry (ALTREMOTE_UPDATE) Megafunction User Guide*.

Remote System Upgrade Registers

The remote system upgrade block contains a series of registers that stores the configuration addresses, watchdog timer settings, and status information. These registers are listed in Table 9–26.

Table 9–26. Remote System Upgrade Registers

Register	Description
Shift register	This register is accessible by the logic array and allows the update, status, and control registers to be written and sampled by user logic. Write access is enabled in remote update mode for factory configurations to allow writes to the update register. Write access is disabled for all application configurations in remote update mode.
Control register	This register contains the current configuration address, the user watchdog timer settings, one option bit for checking early CONF_DONE, and one option bit for selecting the internal oscillator as the startup state machine clock. During a read operation in an application configuration, this register is read into the shift register. When a reconfiguration cycle is started, the contents of the update register are written into the control register.
Update register	This register contains data similar to that in the control register. However, it can only be updated by the factory configuration by shifting data into the shift register and issuing an update operation. When a reconfiguration cycle is triggered by the factory configuration, the control register is updated with the contents of the update register. During a read in a factory configuration, this register is read into the shift register.
Status register	This register is written to by the remote system upgrade circuitry on every reconfiguration to record the cause of the reconfiguration. This information is used by the factory configuration to determine the appropriate action following a reconfiguration. During a capture cycle, this register is read into the shift register.

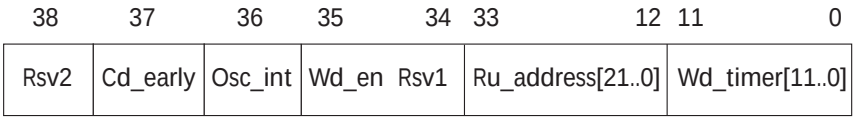
The control and status registers of the remote system upgrade are clocked by the 10-MHz internal oscillator (the same oscillator that controls the user watchdog timer). However, the shift and update registers of the remote system upgrade are clocked by the maximum frequency of 40-MHz user clock input (RU_CLK). There is no minimum frequency for RU_CLK.

Remote System Upgrade Control Register

The remote system upgrade control register stores the application configuration address, the user watchdog timer settings, and option bits for application configuration. In remote update mode for the AS configuration scheme, the control register address bits are set to all zeros (24'b0) at power up to load the AS factory configuration. In remote update mode for the AP configuration scheme, the control register address bits are set to 24'h010000 (24'b1 0000 0000 0000 0000) at power up to load the AP default factory configuration. However, for the AP configuration scheme, you can change the default factory configuration address to any desired address using the APFC_BOOT_ADDR JTAG instruction. Additionally, a factory configuration in remote update mode has write access to this register.

The control register bit positions are shown in Figure 9–36 and listed in Table 9–27. In the figure, the numbers show the bit position of a setting in a register. For example, bit number 35 is the enable bit for the watchdog timer.

Figure 9–36. Remote System Upgrade Control Register



When enabled, the early CONF_DONE check (Cd_early) option bit ensures that there is a valid configuration at the boot address specified by the factory configuration and that it is of the proper size. If an invalid configuration is detected or CONF_DONE pin is asserted too early, the device resets and then reconfigures the factory configuration image. The internal oscillator, as startup state machine clock (Osc_int) option bit, ensures a functional startup clock to eliminate the hanging of startup when enabled. When all option bits are turned on, they provide complete coverage for the programming and startup portions of the application configuration. It is strongly recommended that you turn on both the Cd_early and Osc_int option bits.

☞ The Cd_early and Osc_int option bits for the application configuration must be turned on by the factory configuration.

Table 9–27. Remote System Upgrade Control Register Contents

Control Register Bit	Value	Definition
Wd_timer[11..0]	12'b000000000000	User watchdog time-out value (most significant 12 bits of 29-bit count value: {Wd_timer[11..0],17'b1000})
Ru_address[21..0]	22'b0000000000000000000000	Configuration address (most significant 22 bits of 24-bit boot address value: boot_address[23:0] = {Ru_address[21..0],2'b0})
Rsv1	1'b0	Reserved bit
Wd_en	1'b1	User watchdog timer enable bit
Osc_int *(1)*	1'b1	Internal oscillator as startup state machine clock enable bit
Cd_early *(1)*	1'b1	Early CONF_DONE check
Rsv2	1'b1	Reserved bit

Note to Table 9–27:

(1) Option bit for the application configuration.

Remote System Upgrade Status Register

The remote system upgrade status register specifies the reconfiguration trigger condition. The various trigger and error conditions include:

- Cyclical redundancy check (CRC) error during application configuration
- nSTATUS assertion by an external device due to an error
- Cyclone III device family logic array triggered a reconfiguration cycle, possibly after downloading a new application configuration image

- External configuration reset (nCONFIG) assertion
- User watchdog timer time out

Table 9–28 lists the contents of the current state logic in the status register, when the remote system upgrade master state machine is in factory configuration or application configuration accessing the factory information or application information respectively, and the MSEL pin setting is set to AS or AP configuration scheme. The status register bit in Table 9–28 lists the bit positions in a 32-bit logic.

Table 9–28. Remote System Upgrade Current State Logic Contents In Status Register *(Note 1)*

Current State Logic	Status Register Bit	Definition	Description
Factory information *(2)*	31:30	Master State Machine current state	The current state of the RSU master state machine
	29:24	Reserved bits	Padding bits that are set to all 0's
	23:0	Boot address	The current 24-bit boot address that was used by the configuration scheme as the start address to load the current configuration.
Application information part 1 *(3)*	31:30	Master State Machine current state	The current state of the RSU master state machine
	29	User watchdog timer enable bit	The current state of the user watchdog enable, which is active high
	28:0	User watchdog timer time-out value	The current entire 29-bit watchdog time-out value
Application information part 2 *(3)*	31:30	Master State Machine current state	The current state of the RSU master state machine
	29:24	Reserved bits	Padding bits that are set to all 0's
	23:0	Boot address	The current 24-bit boot address that was used by the configuration scheme as the start address to load the current configuration

Notes to Table 9–28:

(1) The MSEL pin setting is in the AS or AP configuration scheme.
(2) The RSU master state machine is in factory configuration.
(3) The RSU master state machine is in application configuration.

The previous two application configurations are available in the previous state registers (previous state register 1 and previous state register 2), but only for debugging purposes.

Table 9–29 lists the contents of the previous state register 1 and previous state register 2 in the status register when the MSEL pin setting is set to the AS or AP scheme. The status register bit in Table 9–29 shows the bit positions in a 31-bit register. The previous state register 1 and previous state register 2 have the same bit definitions. The previous state register 1 reflects the current application configuration and the previous state register 2 reflects the previous application configuration.

Table 9–29. Remote System Upgrade Previous State Register 1 and Previous State Register 2 Contents in Status Register *(Note 1)*

Status Register Bit	Definition	Description
30	nCONFIG source	One-hot, active-high field that describes the reconfiguration source that caused the Cyclone III device family to leave the previous application configuration. If there is a tie, the higher bit order indicates precedence. For example, if nCONFIG and remote system upgrade nCONFIG reach the reconfiguration state machine at the same time, the nCONFIG precedes the remote system upgrade nCONFIG.
29	CRC error source	
28	nSTATUS source	
27	User watchdog timer source	
26	Remote system upgrade nCONFIG source	
25:24	Master state machine current state	The state of the master state machine during reconfiguration causes the Cyclone III device family to leave the previous application configuration.
23:0	Boot address	The address used by the configuration scheme to load the previous application configuration.

Note to Table 9–29:

(1) The MSEL pin settings are in the AS configuration scheme.

If a capture is inappropriately done, for example, capturing a previous state before the system has entered remote update application configuration for the first time, a value will output from the shift register to indicate that the capture was incorrectly called.

Remote System Upgrade State Machine

The remote system upgrade control and update registers have identical bit definitions, but serve different roles (Table 9–26 on page 9–83). While both registers can only be updated when the device is loaded with a factory configuration image, the update register writes are controlled by the user logic, and the control register writes are controlled by the remote system upgrade state machine.

In factory configurations, the user logic should send the option bits (Cd_early and Osc_int), the configuration address, and watchdog timer settings for the next application configuration bit to the update register. When the logic array configuration reset (RU_nCONFIG) goes high, the remote system upgrade state machine updates the control register with the contents of the update register and starts system reconfiguration from the new application page.

To ensure the successful reconfiguration between the pages, assert RU_nCONFIG signal for a minimum of 250 ns. This is equivalent to strobing the reconfig input of the ALTREMOTE_UPDATE megafunction high for a minimum of 250 ns.

If there is an error or reconfiguration trigger condition, the remote system upgrade state machine directs the system to load a factory or application configuration (based on mode and error condition) by setting the control register accordingly.

Table 9–30 lists the contents of the control register after such an event occurs for all possible error or trigger conditions.

The remote system upgrade status register is updated by the dedicated error monitoring circuitry after an error condition but before the factory configuration is loaded.

Table 9–30. Control Register Contents After an Error or Reconfiguration Trigger Condition

Reconfiguration Error/Trigger	Control Register Setting In Remote Update
nCONFIG reset	All bits are 0
nSTATUS error	All bits are 0
CORE triggered reconfiguration	Update register
CRC error	All bits are 0
Wd time out	All bits are 0

User Watchdog Timer

The user watchdog timer prevents a faulty application configuration from stalling the device indefinitely. The system uses the timer to detect functional errors after an application configuration is successfully loaded into the Cyclone III device family.

The user watchdog timer is a counter that counts down from the initial value loaded into the remote system upgrade control register by the factory configuration. The counter is 29-bits wide and has a maximum count value of 2^{29}. When specifying the user watchdog timer value, specify only the most significant 12 bits. Remote system upgrade circuitry appends 17'b1000 to form the 29 bits value for the watchdog timer. The granularity of the timer setting is 2^{17} cycles. The cycle time is based on the frequency of the 10-MHz internal oscillator.

Table 9–31 lists the operating range of the 10-MHz internal oscillator.

Table 9–31. 10-MHz Internal Oscillator Specifications

Minimum	Typical	Maximum	Unit
5	6.5	10	MHz

The user watchdog timer begins counting after the application configuration enters device user mode. This timer must be periodically reloaded or reset by the application configuration before the timer expires by asserting RU_nRSTIMER. If the application configuration does not reload the user watchdog timer before the count expires, a time-out signal is generated by the remote system upgrade dedicated circuitry. The time-out signal tells the remote system upgrade circuitry to set the user watchdog timer status bit (Wd) in the remote system upgrade status register and reconfigures the device by loading the factory configuration.

To allow remote system upgrade dedicated circuitry to reset the watchdog timer, you must assert the RU_nRSTIMER signal active for a minimum of 250 ns. This is equivalent to strobing the reset_timer input of the ALTREMOTE_UPDATE megafunction high for a minimum of 250 ns.

The user watchdog timer is not enabled during the configuration cycle of the device. Errors during configuration are detected by the CRC engine. Also, the timer is disabled for factory configuration. Functional errors must not exist in the factory configuration because it is stored and validated during production and is never updated remotely.

The user watchdog timer is disabled in factory configurations and during the configuration cycle of the application configuration. It is enabled after the application configuration enters user mode.

Quartus II Software Support

Implementation in your design requires a remote system upgrade interface between the Cyclone III device family logic array and the remote system upgrade circuitry. You must also generate configuration files for production and remote programming of the system configuration memory. The Quartus II software provides these features.

The two implementation options, the ALTREMOTE_UPDATE megafunction and the remote system upgrade atom, are for the interface between the remote system upgrade circuitry and the device logic array interface. Using the megafunction block instead of creating your own logic saves design time and offers more efficient logic synthesis and device implementation.

For more information about the ALTREMOTE_UPDATE megafunction, refer to the *Remote Update Circuitry (ALTREMOTE_UPDATE) Megafunction User Guide*.

Chapter Revision History

Table 9–32 lists the revision history for this chapter.

Table 9–32. Chapter Revision History

Date	Version	Changes Made
December 2009	1.2	■ Updated Table 9–7, Table 9–10, Table 9–22, and Table 9–28. ■ Updated Figure 9–23 and Figure 9–30. ■ Updated the "Programming Serial Configuration Devices" and "Security Against Tampering" sections. ■ Minor changes to the text.
July 2009	1.1	Made a minor correction to the part number.
June 2009	1.0	Initial release.

13. Configuring Cyclone II Devices

CII51013-3.1

Introduction

Cyclone® II devices use SRAM cells to store configuration data. Since SRAM memory is volatile, configuration data must be downloaded to Cyclone II devices each time the device powers up. You can use the active serial (AS) configuration scheme, which can operate at a DCLK frequency up to 40 MHz, to configure Cyclone II devices. You can also use the passive serial (PS) and Joint Test Action Group (JTAG)-based configuration schemes to configure Cyclone II devices. Additionally, Cyclone II devices can receive a compressed configuration bitstream and decompress this data on-the-fly, reducing storage requirements and configuration time.

This chapter explains the Cyclone II configuration features and describes how to configure Cyclone II devices using the supported configuration schemes. This chapter also includes configuration pin descriptions and the Cyclone II configuration file format.

For more information on setting device configuration options or creating configuration files, see the *Software Settings* chapter in the *Configuration Handbook*.

Cyclone II Configuration Overview

You can use the AS, PS, and JTAG configuration schemes to configure Cyclone II devices. You can select which configuration scheme to use by driving the Cyclone II device MSEL pins either high or low as shown in Table 13–1. The MSEL pins are powered by the V_{CCIO} power supply of the bank they reside in. The MSEL[1..0] pins have 9-kΩ internal pull-down resistors that are always active. During power-on reset (POR) and reconfiguration, the MSEL pins have to be at LVTTL V_{IL} or V_{IH} levels to be considered a logic low or logic high, respectively. Therefore, to avoid any problems with detecting an incorrect configuration scheme, you should connect the MSEL[] pins to the V_{CCIO} of the I/O bank they reside in and GND without any pull-up or pull-down resistors. The MSEL[] pins should not be driven by a microprocessor or another device.

Table 13–1. Cyclone II Configuration Schemes		
Configuration Scheme	**MSEL1**	**MSEL0**
AS (20 MHz)	0	0
PS	0	1
Fast AS (40 MHz) *(1)*	1	0
JTAG-based Configuration *(2)*	*(3)*	*(3)*

Notes to Table 13–1:

(1) Only the EPCS16 and EPCS64 devices support a DCLK up to 40 MHz clock; other EPCS devices support a DCLK up to 20 MHz. Refer to the *Serial Configuration Devices Data Sheet* for more information.

(2) JTAG-based configuration takes precedence over other configuration schemes, which means MSEL pin settings are ignored.

(3) Do not leave the MSEL pins floating; connect them to V_{CCIO} or ground. These pins support the non-JTAG configuration scheme used in production. If you are only using JTAG configuration, you should connect the MSEL pins to ground.

You can download configuration data to Cyclone II FPGAs with the AS, PS, or JTAG interfaces using the options in Table 13–2.

Table 13–2. Cyclone II Device Configuration Schemes	
Configuration Scheme	**Description**
AS configuration	Configuration using serial configuration devices (EPCS1, EPCS4, EPCS16 or EPCS64 devices)
PS configuration	Configuration using enhanced configuration devices (EPC4, EPC8, and EPC16 devices), EPC2 and EPC1 configuration devices, an intelligent host (microprocessor), or a download cable
JTAG-based configuration	Configuration via JTAG pins using a download cable, an intelligent host (microprocessor), or the Jam™ Standard Test and Programming Language (STAPL)

Configuration File Format

Table 13–3 shows the approximate uncompressed configuration file sizes for Cyclone II devices. To calculate the amount of storage space required for multiple device configurations, add the file size of each device together.

Table 13–3. Cyclone II Raw Binary File (.rbf) Sizes *Note (1)*

Device	Data Size (Bits)	Data Size (Bytes)
EP2C5	1,265,792	152,998
EP2C8	1,983,536	247,974
EP2C15	3,892,496	486,562
EP2C20	3,892,496	486,562
EP2C35	6,858,656	857,332
EP2C50	9,963,392	1,245,424
EP2C70	14,319,216	1,789,902

Note to Table 13–3:
(1) These values are preliminary.

Use the data in Table 13–3 only to estimate the file size before design compilation. Different configuration file formats, such as a Hexadecimal (**.hex**) or Tabular Text File (**.ttf**) format, have different file sizes. However, for any specific version of the Quartus® II software, any design targeted for the same device has the same uncompressed configuration file size. If compression is used, the file size can vary after each compilation since the compression ratio is dependent on the design.

Configuration Data Compression

Cyclone II devices support configuration data decompression, which saves configuration memory space and time. This feature allows you to store compressed configuration data in configuration devices or other memory and transmit this compressed bitstream to Cyclone II devices. During configuration, the Cyclone II device decompresses the bitstream in real time and programs its SRAM cells.

Preliminary data indicates that compression reduces configuration bitstream size by 35 to 55%.

Cyclone II devices support decompression in the AS and PS configuration schemes. Decompression is not supported in JTAG-based configuration.

Although they both use the same compression algorithm, the decompression feature supported by Cyclone II devices is different from the decompression feature in enhanced configuration devices (EPC16, EPC8, and EPC4 devices). The data decompression feature in the enhanced configuration devices allows them to store compressed data and decompress the bitstream before transmitting it to the target devices.

In PS mode, you should use the Cyclone II decompression feature since sending compressed configuration data reduces configuration time. You should not use both the Cyclone II device and the enhanced configuration device decompression features simultaneously. The compression algorithm is not intended to be recursive and could expand the configuration file instead of compressing it further.

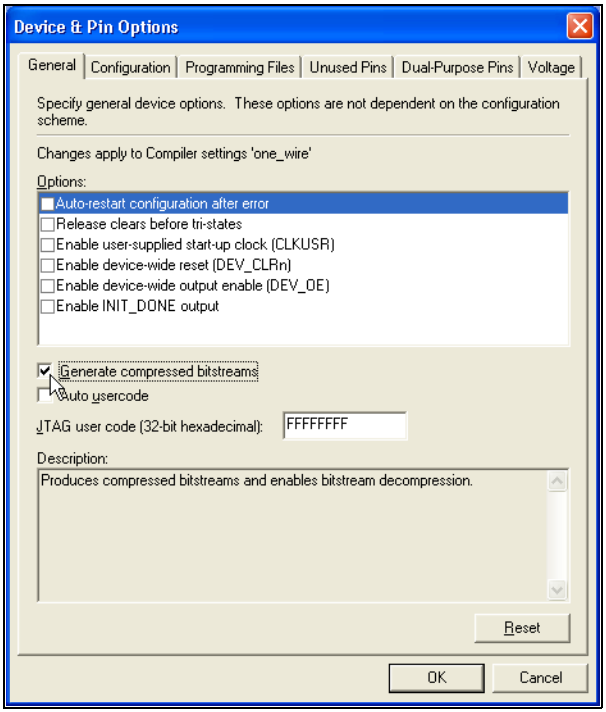
You should use the Cyclone II decompression feature during AS configuration if you need to save configuration memory space in the serial configuration device.

When you enable compression, the Quartus II software generates configuration files with compressed configuration data. This compressed file reduces the storage requirements in the configuration device or flash, and decreases the time needed to transmit the bitstream to the Cyclone II device. The time required by a Cyclone II device to decompress a configuration file is less than the time needed to transmit the configuration data to the FPGA.

There are two methods to enable compression for Cyclone II bitstreams: before design compilation (in the Compiler Settings menu) and after design compilation (in the **Convert Programming Files** window).

To enable compression in the project's compiler settings, select **Device** under the Assignments menu to bring up the settings window. After selecting your Cyclone II device open the **Device & Pin Options** window, and in the **General settings** tab enable the check box for **Generate compressed bitstreams** (see Figure 13–1).

Figure 13–1. Enabling Compression for Cyclone II Bitstreams in Compiler Settings

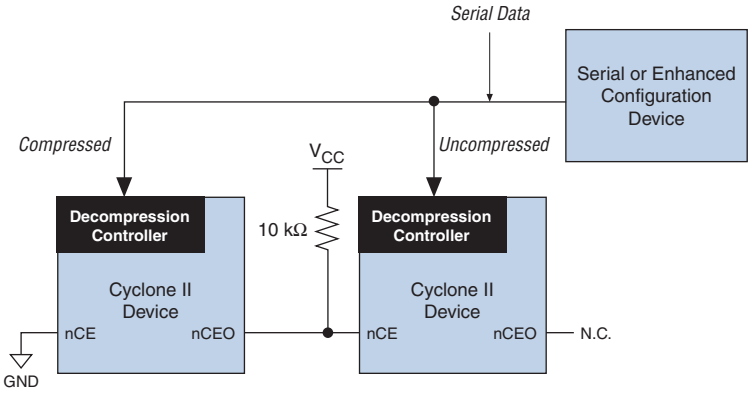


You can also use the following steps to enable compression when creating programming files from the Convert Programming Files window.

1. Click **Convert Programming Files** (File menu).
2. Select the Programming File type. Only Programmer Object Files (**.pof**), SRAM HEXOUT, RBF, or TTF files support compression.
3. For POFs, select a configuration device.
4. Select **Add File** and add a Cyclone II SRAM Object File(s) (**.sof**).
5. Select the name of the file you added to the SOF Data area and click on **Properties**.
6. Check the **Compression** check box.

When multiple Cyclone II devices are cascaded, the compression feature can be selectively enabled for each device in the chain. Figure 13–2 depicts a chain of two Cyclone II devices. The first Cyclone II device has compression enabled and therefore receives a compressed bitstream from the configuration device. The second Cyclone II device has the compression feature disabled and receives uncompressed data.

Figure 13–2. Compressed & Uncompressed Configuration Data in a Programming File



You can generate programming files (for example, POF files) for this setup in the Quartus II software.

Active Serial Configuration (Serial Configuration Devices)

In the AS configuration scheme, Cyclone II devices are configured using a serial configuration device. These configuration devices are low-cost devices with non-volatile memory that feature a simple, four-pin interface and a small form factor. These features make serial configuration devices an ideal low-cost configuration solution.

For more information on serial configuration devices, see the *Serial Configuration Devices Data Sheet* in the Configuration Handbook.

Serial configuration devices provide a serial interface to access configuration data. During device configuration, Cyclone II devices read configuration data via the serial interface, decompress data if necessary, and configure their SRAM cells. The FPGA controls the configuration interface in the AS configuration scheme, while the external host (e.g., the configuration device or microprocessor) controls the interface in the PS configuration scheme.

The Cyclone II decompression feature is available when configuring your Cyclone II device using AS mode.

Table 13–4 shows the MSEL pin settings when using the AS configuration scheme.

Table 13–4. Cyclone II Configuration Schemes

Configuration Scheme	MSEL1	MSEL0
AS (20 MHz)	0	0
Fast AS (40 MHz) *(1)*	1	0

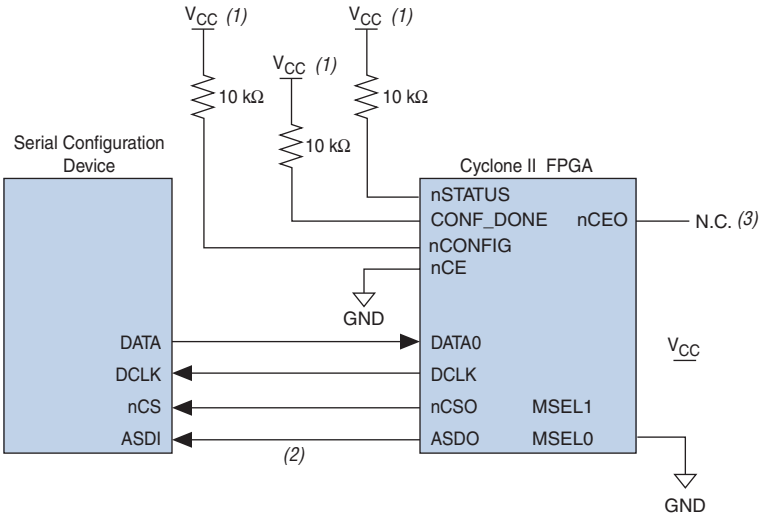
Note to Table 13–4:

(1) Only the EPCS16 and EPCS64 devices support a DCLK up to 40 MHz clock; other EPCS devices support a DCLK up to 20 MHz. Refer to the *Serial Configuration Devices Data Sheet* for more information.

Single Device AS Configuration

Serial configuration devices have a four-pin interface: serial clock input (DCLK), serial data output (DATA), AS data input (ASDI), and an active-low chip select (nCS). This four-pin interface connects to Cyclone II device pins, as shown in Figure 13–3.

Figure 13–3. Single Device AS Configuration



Notes to Figure 13–3:

(1) Connect the pull-up resistors to a 3.3-V supply.

(2) Cyclone II devices use the ASDO to ASDI path to control the configuration device.

(3) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed another device's nCE pin.

Upon power-up, the Cyclone II device goes through a POR. During POR, the device resets, holds nSTATUS and CONF_DONE low, and tri-states all user I/O pins. After POR, which typically lasts 100 ms, the Cyclone II device releases nSTATUS and enters configuration mode when the external 10-kΩ resistor pulls the nSTATUS pin high. Once the FPGA successfully exits POR, all user I/O pins continue to be tri-stated. Cyclone II devices have weak pull-up resistors on the user I/O pins which are on before and during configuration.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration are available in the *DC Characteristics & Timing Specifications* chapter of the *Cyclone II Device Handbook*.

The configuration cycle consists of the reset, configuration, and initialization stages.

Reset Stage

When nCONFIG or nSTATUS are low, the device is in reset. After POR, the Cyclone II device releases nSTATUS. An external 10-kΩ pull-up resistor pulls the nSTATUS signal high, and the Cyclone II device enters configuration mode.

V_{CCINT} and V_{CCIO} of the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

Configuration Stage

The serial clock (DCLK) generated by the Cyclone II device controls the entire configuration cycle and provides the timing for the serial interface. Cyclone II devices use an internal oscillator to generate DCLK. Using the MSEL[] pins, you can select either a 20- or 40-MHz oscillator. Although you can select either 20- or 40-MHz oscillator when designing with serial configuration devices, the 40-MHz oscillator provides faster configuration times. There is some variation in the internal oscillator frequency because of the process, temperature, and voltage conditions in Cyclone II devices. The internal oscillator is designed such that its maximum frequency is guaranteed to meet EPCS device specifications.

Table 13–5 shows the AS DCLK output frequencies.

Table 13–5. AS DCLK Output Frequency *Note (1)*

Oscillator Selected	Minimum	Typical	Maximum	Units
40 MHz	20	26	40	MHz
20 MHz	10	13	20	MHz

Note to Table 13–5:
(1) These values are preliminary.

In both AS and Fast AS configuration schemes, the serial configuration device latches input and control signals on the rising edge of DCLK and drives out configuration data on the falling edge. Cyclone II devices drive out control signals on the falling edge of DCLK and latch configuration data on the falling edge of DCLK.

In configuration mode, the Cyclone II device enables the serial configuration device by driving its nCSO output pin low, which connects to the chip select (nCS) pin of the configuration device. The Cyclone II device uses the serial clock (DCLK) and serial data output (ASDO) pins to send operation commands and/or read address signals to the serial

configuration device. The configuration device then provides data on its serial data output (DATA) pin, which connects to the DATA0 input of the Cyclone II device.

After the Cyclone II device receives all the configuration bits, it releases the open-drain CONF_DONE pin, which is then pulled high by an external 10-kΩ resistor. Also, the Cyclone II device stops driving the DCLK signal. Initialization begins only after the CONF_DONE signal reaches a logic high level. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize. All AS configuration pins (DATA0, DCLK, nCSO, and ASDO) have weak internal pull-up resistors which are always active. After configuration, these pins are set as input tri-stated and are pulled high by the internal weak pull-up resistors.

Initialization Stage

In Cyclone II devices, the initialization clock source is either the Cyclone II 10-MHz (typical) internal oscillator (separate from the AS internal oscillator) or the optional CLKUSR pin. The internal oscillator is the default clock source for initialization. If the internal oscillator is used, the Cyclone II device provides itself with enough clock cycles for proper initialization. The advantage of using the internal oscillator is you do not need to send additional clock cycles from an external source to the CLKUSR pin during the initialization stage. Additionally, you can use the CLKUSR pin as a user I/O pin.

If you want to delay the initialization of the device, you can use the CLKUSR pin option. Using the CLKUSR pin allows you to control when your device enters user mode. The device can be delayed from entering user mode for an indefinite amount of time. When you enable the **User Supplied Start-Up Clock** option, the CLKUSR pin is the initialization clock source. Supplying a clock on CLKUSR does not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, Cyclone II devices require 299 clock cycles to initialize properly and support a CLKUSR f_{MAX} of 100 MHz.

Cyclone II devices offer an optional INIT_DONE pin which signals the end of initialization and the start of user mode with a low-to-high transition. The **Enable INIT_DONE output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** window. If you use the INIT_DONE pin, an external 10-kΩ pull-up resistor is required to pull the signal high when nCONFIG is low and during the beginning of configuration. Once the optional bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. This low-to-high transition signals that the FPGA has entered user mode. If you do not use the INIT_DONE pin, the initialization period is complete after CONF_DONE goes high and 299 clock cycles are sent to the CLKUSR pin or after the time t_{CF2UM} (see Table 13–8) if the Cyclone II device uses the internal oscillator.

User Mode

When initialization is complete, the FPGA enters user mode. In user mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

When the Cyclone II device is in user mode, you can initiate reconfiguration by pulling the nCONFIG signal low. The nCONFIG signal should be low for at least 2 µs. When nCONFIG is pulled low, the Cyclone II device is reset and enters the reset stage. The Cyclone II device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Once nCONFIG returns to a logic high level and nSTATUS is released by the Cyclone II device, reconfiguration begins.

Error During Configuration

If an error occurs during configuration, the Cyclone II device drives the nSTATUS signal low to indicate a data frame error, and the CONF_DONE signal stays low. If you enable the **Auto-restart configuration after error** option in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box, the Cyclone II device resets the serial configuration device by pulsing nCSO, releases nSTATUS after a reset time-out period (about 40 µs), and retries configuration. If the **Auto-restart configuration after error** option is turned off, the external system must monitor nSTATUS for errors and then pull nCONFIG low for at least 2 µs to restart configuration.

If you use the optional CLKUSR pin and the nCONFIG pin is pulled low to restart configuration during device initialization, ensure CLKUSR continues to toggle during the time nSTATUS is low (a maximum of 40 µs).

For more information on configuration issues, see the *Debugging Configuration Problems* chapter of the *Configuration Handbook* and the FPGA Configuration Troubleshooter on the Altera web site (**www.altera.com**).

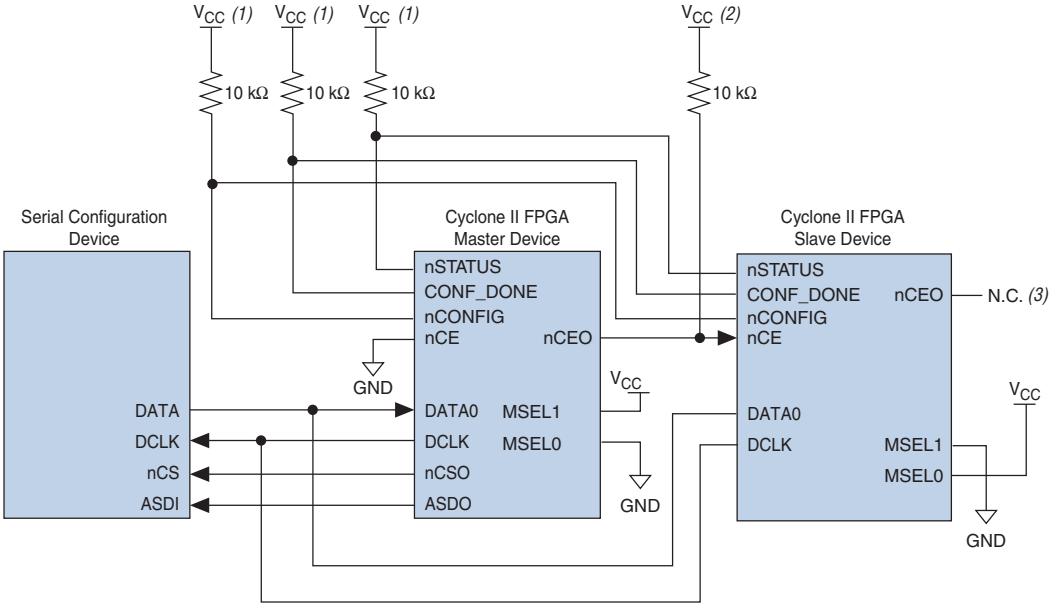
Multiple Device AS Configuration

You can configure multiple Cyclone II devices using a single serial configuration device. You can cascade multiple Cyclone II devices using the chip-enable (nCE) and chip-enable-out (nCEO) pins. Connect the nCE pin of the first device in the chain to ground and connect the nCEO pin to the nCE pin of the next device in the chain. Use an external 10-kΩ pull-up resistor to pull the nCEO signal high to its V_{CCIO} level to help the internal weak pull-up resistor. When the first device captures all of its configuration data from the bitstream, it transitions its nCEO pin low, initiating the configuration of the next device in the chain. You can leave the nCEO pin of the last device unconnected or use it as a user I/O pin after configuration if the last device in chain is a Cyclone II device.

The Quartus II software sets the Cyclone II device nCEO pin as an output pin driving to ground by default. If the device is in a chain, and the nCEO pin is connected to the next device's nCE pin, you must make sure that the nCEO pin is not used as a user I/O pin after configuration. The software setting is in the **Dual-Purpose Pins** tab of the **Device & Pin Options** dialog box in Quartus II software.

The first Cyclone II device in the chain is the configuration master and controls the configuration of the entire chain. Select the AS configuration scheme for the first Cyclone II device and the PS configuration scheme for the remaining Cyclone II devices (configuration slaves). Any other Altera® device that supports PS configuration can also be part of the chain as a configuration slave. In a multiple device chain, the nCONFIG, nSTATUS, CONF_DONE, DCLK, and DATA0 pins of each device in the chain are connected (see Figure 13–4). Figure 13–4 shows the pin connections for this setup.

Figure 13–4. Multiple Device AS Configuration



Notes to Figure 13–4:

(1) Connect the pull-up resistors to a 3.3-V supply.

(2) Connect the pull-up resistor to the V_{CCIO} supply voltage of I/O bank that the nCEO pin resides in.

(3) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed another device's nCE pin.

As shown in Figure 13–4, the nSTATUS and CONF_DONE pins on all target FPGAs are connected together with external pull-up resistors. These pins are open-drain bidirectional pins on the FPGAs. When the first device asserts nCEO (after receiving all of its configuration data), it releases its CONF_DONE pin. However, the subsequent devices in the chain keep the CONF_DONE signal low until they receive their configuration data. When all the target FPGAs in the chain have received their configuration data and have released CONF_DONE, the pull-up resistor pulls this signal high, and all devices simultaneously enter initialization mode.

During initialization, the initialization clock source is either the Cyclone II 10 MHz (typical) internal oscillator (separate from the AS internal oscillator) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Cyclone II device provides itself with enough clock cycles for proper initialization. The advantage of using the internal oscillator is you do not need to send additional clock cycles from an external source to the CLKUSR pin during the initialization stage. You can also make use of the CLKUSR pin as a user I/O pin, which means you have an additional user I/O pin.

If you want to delay the initialization of the devices in the chain, you can use the CLKUSR pin option. The CLKUSR pin allows you to control when your device enters user mode. This feature also allows you to control the order of when each device enters user mode by feeding a separate clock to each device's CLKUSR pin. By using the CLKUSR pins, you can choose any device in the multiple device chain to enter user mode first and have the other devices enter user mode at a later time.

Different device families may require a different number of initialization clock cycles. Therefore, if your multiple device chain consists of devices from different families, the devices may enter user mode at a slightly different time due to the different number of initialization clock cycles required. However, if the number of initialization clock cycles is similar across different device families or if the devices are from the same family, then the devices enter user mode at the same time. See the respective device family handbook for more information about the number of initialization clock cycles required.

If an error occurs at any point during configuration, the FPGA with the error drives the nSTATUS signal low. If you enable the **Auto-restart configuration after error** option, the entire chain begins reconfiguration after a reset time-out period (a maximum of 40 µs). If the **Auto-restart configuration after error** option is turned off, a microprocessor or controller must monitor nSTATUS for errors and then pulse nCONFIG low to restart configuration. The microprocessor or controller can pulse nCONFIG if it is under system control rather than tied to V_{CC}.

While you can cascade Cyclone II devices, serial configuration devices cannot be cascaded or chained together.

If you use the optional CLKUSR pin and the nCONFIG is pulled low to restart configuration during device initialization, make sure the CLKUSR pin continues to toggle while nSTATUS is low (a maximum of 40 µs).

If the configuration bitstream size exceeds the capacity of a serial configuration device, you must select a larger configuration device and/or enable the compression feature. When configuring multiple devices, the size of the bitstream is the sum of the individual devices' configuration bitstreams.

Configuring Multiple Cyclone II Devices with the Same Design

Certain designs require you to configure multiple Cyclone II devices with the same design through a configuration bitstream or SOF. You can do this through one of two methods, as described in this section. For both methods, the serial configuration devices cannot be cascaded or chained together.

Multiple SOFs

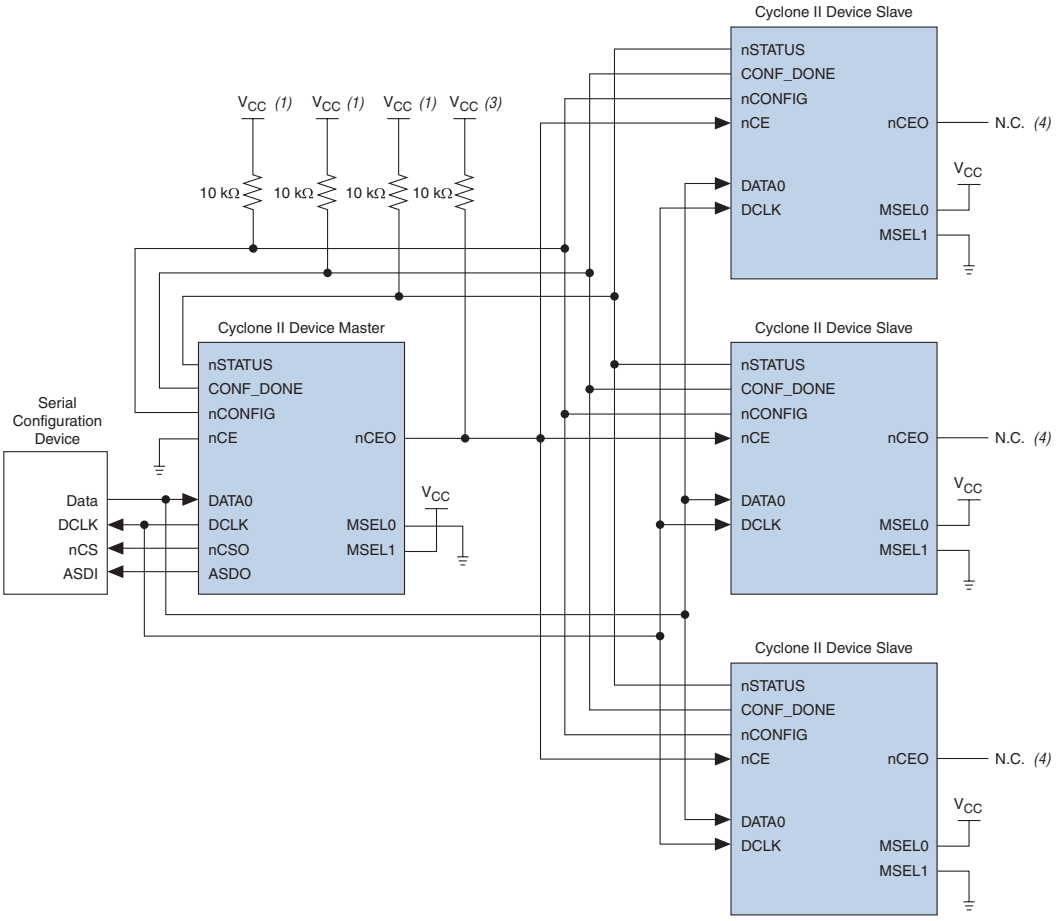
In the first method, two copies of the SOF file are stored in the serial configuration device. Use the first copy to configure the master Cyclone II device and the second copy to configure all remaining slave devices concurrently. In this setup, the master Cyclone II device is in AS mode, and the slave Cyclone II devices are in PS mode (MSEL=01). See Figure 13–5.

To configure four identical Cyclone II devices with the same SOF file, connect the three slave devices for concurrent configuration as shown in Figure 13–5. The nCEO pin from the master device drives the nCE input pins on all three slave devices. Connect the configuration device's DATA and DCLK pins to the Cyclone II device's DATA and DCLK pins in parallel. During the first configuration cycle, the master device reads its configuration data from the serial configuration device while holding nCEO high. After completing its configuration cycle, the master drives nCE low and transmits the second copy of the configuration data to all three slave devices, configuring them simultaneously.

The advantage of using the setup in Figure 13–5 is that you can have a different SOF file for the Cyclone II master device. However, all the Cyclone II slave devices must be configured with the same SOF file. The SOF files in this configuration method can be either compressed or uncompressed.

You can still use this method if the master and slave Cyclone II devices use the same SOF.

Figure 13–5. Multiple Device AS Configuration When FPGAs Receive the Same Data with Multiple SOFs



Notes to Figure 13–5:

(1) Connect the pull-up resistors to a 3.3-V supply.

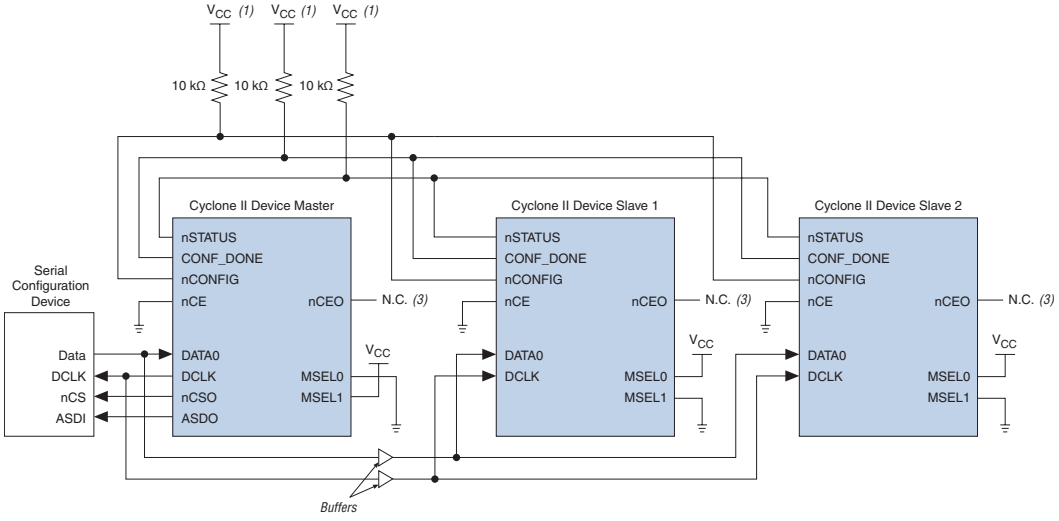
(2) Connect the pull-up resistor to the V_{CCIO} supply voltage of I/O bank that the nCEO pin resides in.

(3) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed another device's nCE pin.

Single SOF

The second method configures both the master and slave Cyclone II devices with the same SOF. The serial configuration device stores one copy of the SOF file. This setup is shown in Figure 13–6 where the master is setup in AS mode, and the slave devices are setup in PS mode (MSEL=01). You could setup one or more slave devices in the chain and all the slave devices are setup in the same way as shown in Figure 13–6.

Figure 13–6. Multiple Device AS Configuration When FPGAs Receive the Same Data with a Single SOF



Notes to Figure 13–6:

(1) Connect the pull-up resistors to a 3.3-V supply.

(2) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed another device's nCE pin.

In this setup, all the Cyclone II devices in the chain are connected for concurrent configuration. This can reduce the AS configuration time because all the Cyclone II devices are configured in one configuration cycle. Connect the nCE input pins of all the Cyclone II devices to ground. You can either leave the nCEO output pins on all the Cyclone II devices unconnected or use the nCEO output pins as normal user I/O pins. The DATA and DCLK pins are connected in parallel to all the Cyclone II devices.

You should put a buffer before the DATA and DCLK output from the master Cyclone II device to avoid signal strength and signal integrity issues. The buffer should not significantly change the DATA-to-DCLK relationships or delay them with respect to other AS signals (ASDI and nCS). Also, the buffer should only drive the slave Cyclone II devices, so that the timing between the master Cyclone II device and serial configuration device is unaffected.

This configuration method supports both compressed and uncompressed SOFs. Therefore, if the configuration bitstream size exceeds the capacity of a serial configuration device, you can enable the compression feature in the SOF file used or you can select a larger serial configuration device.

Estimating AS Configuration Time

The AS configuration time is the time it takes to transfer data from the serial configuration device to the Cyclone II device. The Cyclone II DCLK output (generated from an internal oscillator) clocks this serial interface. As listed in Table 13–5, if you are using the 40-MHz oscillator, the DCLK minimum frequency is 20 MHz (50 ns). Therefore, the maximum configuration time estimate for an EP2C5 device (1,223,980 bits of uncompressed data) is:

RBF size × (maximum DCLK period / 1 bit per DCLK cycle) = estimated maximum configuration time

1,223,980 bits × (50 ns / 1 bit) = 61.2 ms

To estimate the typical configuration time, use the typical DCLK period listed in Table 13–5. With a typical DCLK period of 38.46 ns, the typical configuration time is 47.1 ms. Enabling compression reduces the amount of configuration data that is transmitted to the Cyclone II device, which also reduces configuration time. On average, compression reduces configuration time by 50%.

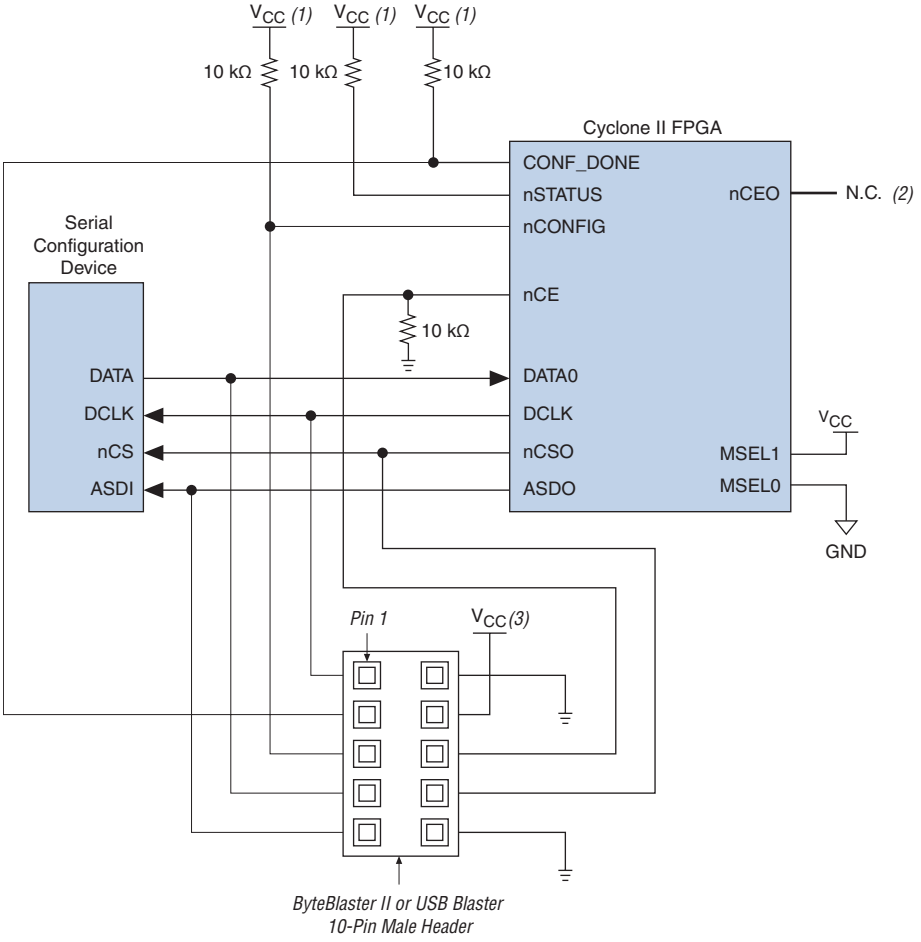
Programming Serial Configuration Devices

Serial configuration devices are non-volatile, flash-memory-based devices. You can program these devices in-system using the USB-Blaster™ or ByteBlaster™ II download cable. Alternatively, you can program them using the Altera Programming Unit (APU), supported third-party programmers, or a microprocessor with the SRunner software driver.

You can use the AS programming interface to program serial configuration devices in-system. During in-system programming, the download cable disables FPGA access to the AS interface by driving the `nCE` pin high. Cyclone II devices are also held in reset by pulling the `nCONFIG` signal low. After programming is complete, the download cable releases the `nCE` and `nCONFIG` signals, allowing the pull-down and pull-up resistor to drive GND and V_{CC}, respectively. Figure 13–7 shows the download cable connections to the serial configuration device.

For more information on the USB-Blaster download cable, see the *USB-Blaster USB Port Download Cable Data Sheet*. For more information on the ByteBlaster II cable, see the *ByteBlaster II Download Cable Data Sheet*.

Figure 13–7. In-System Programming of Serial Configuration Devices



Notes to Figure 13–7:

(1) Connect these pull-up resistors to 3.3-V supply.

(2) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed other device's nCE pin.

(3) Power up the ByteBlaster II or USB Blaster cable's V_{CC} with a 3.3-V supply.

You can use the Quartus II software with the APU and the appropriate configuration device programming adapter to program serial configuration devices. All serial configuration devices are offered in an 8-pin or 16-pin small outline integrated circuit (SOIC) package and can be programmed using the PLMSEPC-8 adapter.

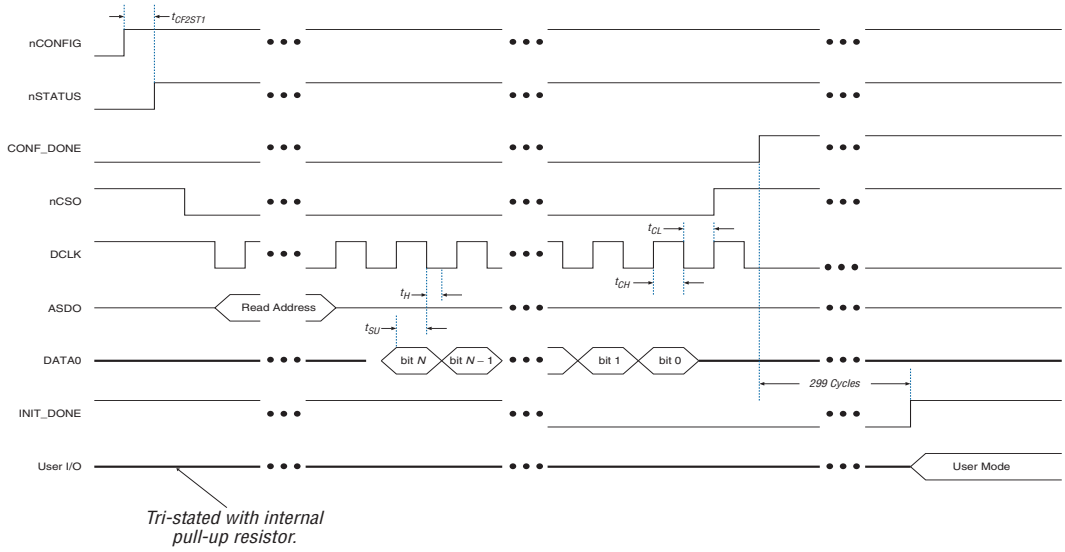
Altera programming hardware (APU) or other third-party programming hardware can be used to program blank serial configuration devices before they are mounted onto PCBs. Alternatively, you can use an on-board microprocessor to program the serial configuration device on the PCB using C-based software drivers provided by Altera (i.e., the SRunner software driver).

A serial configuration device can be programmed in-system by an external microprocessor using SRunner. SRunner is a software driver developed for embedded serial configuration device programming, which can be easily customized to fit in different embedded systems. SRunner can read a Raw Programming Data File (**.rpd**) and write to the serial configuration devices. The serial configuration device programming time using SRunner is comparable to the programming time when using the Quartus II Programmer.

For more information about SRunner, see the *SRunner: An Embedded Solution for EPCS Programming White Paper* and the source code on the Altera web site at www.altera.com. For more information on programming serial configuration devices, see the *Serial Configuration Devices Data Sheet* in the *Configuration Handbook*.

Figure 13–8 shows the timing waveform for the AS configuration scheme using a serial configuration device.

Figure 13–8. AS Configuration Timing



PS Configuration

You can use an Altera configuration device, a download cable, or an intelligent host, such as a MAX® II device or microprocessor to configure a Cyclone II device with the PS scheme. In the PS scheme, an external host (configuration device, MAX II device, embedded processor, or host PC) controls configuration. Configuration data is input to the target Cyclone II devices via the DATA0 pin at each rising edge of DCLK.

The Cyclone II decompression feature is fully available when configuring your Cyclone II device using PS mode.

Table 13–6 shows the MSEL pin settings when using the PS configuration scheme.

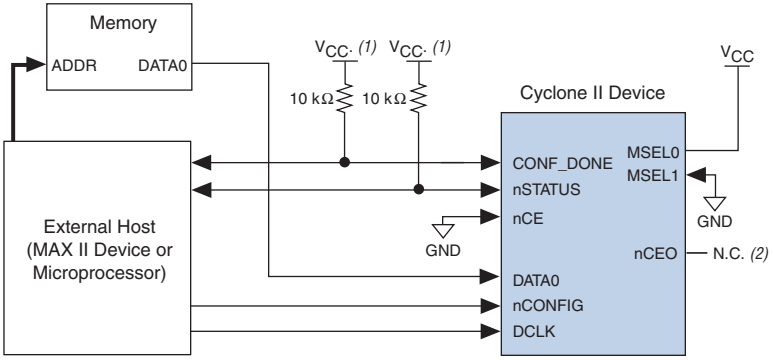
Table 13–6. Cyclone II MSEL Pin Settings for PS Configuration Schemes

Configuration Scheme	MSEL1	MSEL0
PS	0	1

Single Device PS Configuration Using a MAX II Device as an External Host

In the PS configuration scheme, you can use a MAX II device as an intelligent host that controls the transfer of configuration data from a storage device, such as flash memory, to the target Cyclone II device. Configuration data can be stored in RBF, HEX, or TTF format. Figure 13–9 shows the configuration interface connections between the Cyclone II device and a MAX II device for single device configuration.

Figure 13–9. Single Device PS Configuration Using an External Host



Notes to Figure 13–9:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for the device. V_{CC} should be high enough to meet the VIH specification of the I/O on the device and the external host.

(2) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed other device's nCE pin.

Upon power-up, the Cyclone II device goes through a POR, which lasts approximately 100 ms. During POR, the device resets, holds nSTATUS low, and tri-states all user I/O pins. Once the FPGA successfully exits POR, all user I/O pins continue to be tri-stated.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *Cyclone II Device Handbook*.

The configuration cycle consists of three stages: reset, configuration, and initialization.

Reset Stage

While the Cyclone II device's nCONFIG or nSTATUS pins are low, the device is in reset. To initiate configuration, the MAX II device must transition the Cyclone II nCONFIG pin from low to high.

V_{CCINT} and V_{CCIO} of the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When the Cyclone II nCONFIG pin transitions high, the Cyclone II device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. Once nSTATUS is released, the FPGA is ready to receive configuration data and the MAX II device can start the configuration at any time.

Configuration Stage

After the Cyclone II device's nSTATUS pin transitions high, the MAX II device should send the configuration data on the DATA0 pin one bit at a time. If you are using configuration data in RBF, HEX, or TTF format, send the least significant bit (LSB) of each data byte first. For example, if the RBF contains the byte sequence 02 1B EE 01 FA, you should transmit the serial bitstream 0100-0000 1101-1000 0111-0111 1000-0000 0101-1111 to the device first.

The Cyclone II device receives configuration data on its DATA0 pin and the clock on the DCLK pin. Data is latched into the FPGA on the rising edge of DCLK. Data is continuously clocked into the target device until the CONF_DONE pin transitions high. After the Cyclone II device receives all the configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by an external 10-kΩ pull-up resistor. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

The configuration clock (DCLK) speed must be below the specified system frequency (see Table 13–7) to ensure correct configuration. No maximum DCLK period exists, which means you can pause configuration by halting DCLK for an indefinite amount of time.

Initialization Stage

In Cyclone II devices, the initialization clock source is either the Cyclone II internal oscillator (typically 10 MHz) or the optional CLKUSR pin. The internal oscillator is the default clock source for initialization. If you use the internal oscillator, the Cyclone II device makes sure to provide enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device. You do not need to provide additional clock cycles externally during the initialization stage. Driving DCLK back to the device after configuration is complete does not affect device operation. Additionally, if you use the internal oscillator as the clock source, you can use the CLKUSR pin as a user I/O pin.

If you want to delay the initialization of the device, you can use the CLKUSR pin. Using the CLKUSR pin allows you to control when your device enters user mode. You can delay the device from entering user mode for an indefinite amount of time.

The **Enable user-supplied start-up clock (CLKUSR)** option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR does not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, Cyclone II devices require 299 clock cycles to initialize properly and support a CLKUSR f_{MAX} of 100 MHz.

If the optional CLKUSR pin is being used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure that CLKUSR continues toggling during the time nSTATUS is low (maximum of 40 µs).

An optional INIT_DONE pin signals the end of initialization and the start of user mode with a low-to-high transition. By default, the INIT_DONE output is disabled. You can enable the INIT_DONE output by turning on the **Enable INIT_DONE output** option in the Quartus II software. If you use the INIT_DONE pin, an external 10-kΩ pull-up resistor pulls the pin high when nCONFIG is low and during the beginning of configuration. Once the optional bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin transitions low. When initialization is complete, the INIT_DONE pin is released and pulled high. The MAX II device must be able to detect this low-to-high transition, which signals the FPGA has entered user mode.

If you want to use the INIT_DONE pin as a user I/O pin, you should wait for the maximum value of t_{CD2UM} (see Table 13–7) after the CONF_DONE signal transitions high so to ensure the Cyclone II device has been initialized properly and is in user mode.

Make sure the MAX II device does not drive the CONF_DONE signal low during configuration, initialization, and before the device enters user mode.

User Mode

When initialization is complete, the Cyclone II device enters user mode. In user mode, the user I/O pins no longer have pull-up resistors and function as assigned in your design.

To ensure DCLK and DATA0 are not left floating at the end of configuration, the MAX II device must drive them either high or low, which ever is convenient on your PCB. The Cyclone II device DATA0 pin is not available as a user I/O pin after configuration.

When the FPGA is in user mode, you can initiate a reconfiguration by transitioning the nCONFIG pin low-to-high. The nCONFIG pin must be low for at least 2 µs. When the nCONFIG transitions low, the Cyclone II

device also pulls nSTATUS and CONF_DONE low and tri-states all I/O pins. Once the nCONFIG pin returns to a logic high level and the Cyclone II device releases the nSTATUS pin, the MAX II device can begin reconfiguration.

Error During Configuration

If an error occurs during configuration, the Cyclone II device transitions its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin tells the MAX II device that there is an error. If you turn on the **Auto-restart configuration after error** option in the Quartus II software, the Cyclone II device releases nSTATUS after a reset time-out period (maximum of 40 µs). After nSTATUS is released and pulled high by a pull-up resistor, the MAX II device can try to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

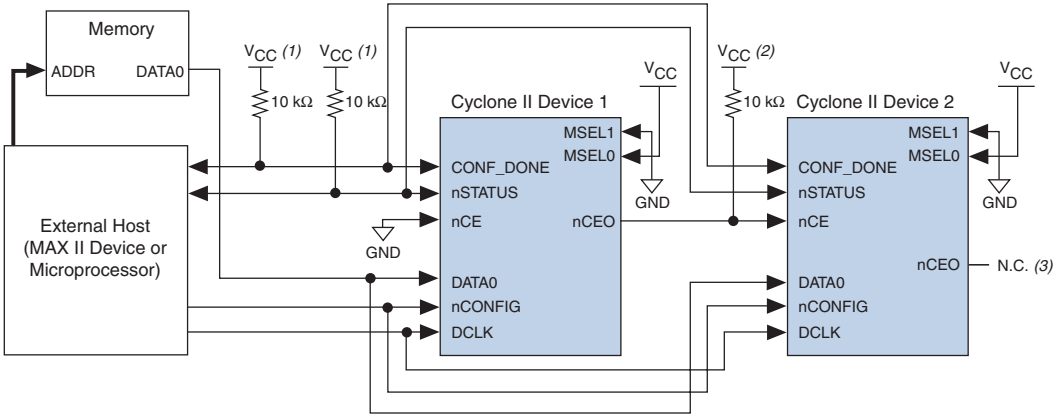
The MAX II device can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The MAX II device must monitor the Cyclone II device's CONF_DONE pin to detect errors and determine when programming completes. If all configuration data is sent, but CONF_DONE or INIT_DONE do not transition high, the MAX II device must reconfigure the target device.

For more information on configuration issues, see the *Debugging Configuration Problems* chapter of the *Configuration Handbook* and the FPGA Configuration Troubleshooter on the Altera web site (**www.altera.com**).

Multiple Device PS Configuration Using a MAX II Device as an External Host

Figure 13–10 shows how to configure multiple devices using a MAX II device. This circuit is similar to the PS configuration circuit for a single device, except Cyclone II devices are cascaded for multiple device configuration.

Figure 13–10. Multiple Device PS Configuration Using an External Host



Notes to Figure 13–10:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain. V_{CC} should be high enough to meet the V_{IH} specification of the I/O on the devices and the external host.

(2) Connect the pull-up resistor to the V_{CCIO} supply voltage of I/O bank that the nCEO pin resides in.

(3) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed another device's nCE pin.

In multiple device PS configuration, connect the first Cyclone II device's nCE pin to GND and connect the nCEO pin to the nCE pin of the next Cyclone II device in the chain. Use an external 10-kΩ pull-up resistor to pull the Cyclone II device's nCEO pin high to its V_{CCIO} level to help the internal weak pull-up resistor when the nCEO pin feeds next Cyclone II device's nCE pin. The input to the nCE pin of the last Cyclone II device in the chain comes from the previous Cyclone II device. After the first device completes configuration in a multiple device configuration chain, its nCEO pin transitions low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration within one clock cycle. Therefore, the MAX II device begins to transfer data to the next Cyclone II device without interruption. The nCEO pin is a dual-purpose pin in Cyclone II devices. You can leave the nCEO pin of the last device unconnected or use it as a user I/O pin after configuration if the last device in chain is a Cyclone II device.

The Quartus II software sets the Cyclone II device nCEO pin as a dedicated output by default. If the nCEO pin feeds the next device's nCE pin, you must make sure that the nCEO pin is not used as a user I/O after configuration. This software setting is in the **Dual-Purpose Pins** tab of the **Device & Pin Options** dialog box in Quartus II software.

You must connect all other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) to every Cyclone II device in the chain. The configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. You should buffer the DCLK and DATA lines for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Since all nSTATUS and CONF_DONE pins are connected, if any Cyclone II device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first Cyclone II detects an error, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single Cyclone II device detecting an error.

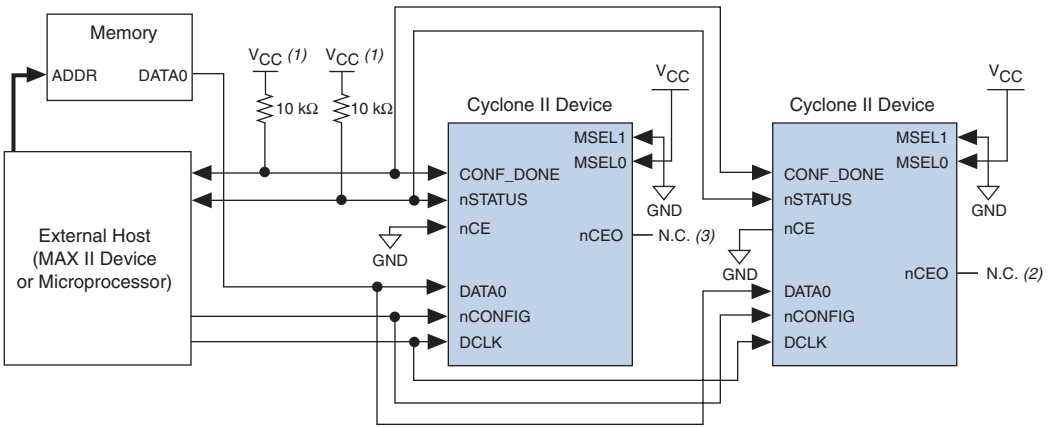
If the **Auto-restart configuration after error** option is turned on, the Cyclone II devices release their nSTATUS pins after a reset time-out period (maximum of 40 µs). After all nSTATUS pins are released and pulled high, the MAX II device reconfigures the chain without pulsing nCONFIG low. If the **Auto-restart configuration after error** option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

If you want to delay the initialization of the devices in the chain, you can use the CLKUSR pin option. The CLKUSR pin allows you to control when your device enters user mode. This feature also allows you to control the order of when each device enters user mode by feeding a separate clock to each device's CLKUSR pin. By using the CLKUSR pins, you can choose any device in the multiple device chain to enter user mode first and have the other devices enter user mode at a later time.

Different device families may require a different number of initialization clock cycles. Therefore, if your multiple device chain consists of devices from different families, the devices may enter user mode at a slightly different time due to the different number of initialization clock cycles required. However, if the number of initialization clock cycles is similar across different device families or if the devices are from the same family, then the devices enter user mode at the same time. See the respective device family handbook for more information about the number of initialization clock cycles required.

If your system has multiple Cyclone II devices (in the same density and package) with the same configuration data, you can configure them in one configuration cycle by connecting all device's nCE pins to ground and connecting all the Cyclone II device's configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) together. You can also use the nCEO pin as a user I/O pin after configuration. The configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Make sure the DCLK and DATA lines are buffered for every fourth device. All devices start and complete configuration at the same time. Figure 13–11 shows multiple device PS configuration when both Cyclone II devices are receiving the same configuration data.

Figure 13–11. Multiple Device PS Configuration When Both FPGAs Receive the Same Data



Notes to Figure 13–11:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain. V_{CC} should be high enough to meet the V_{IH} specification of the I/O on the devices and the external host.

(2) The nCEO pins of both devices can be left unconnected or used as user I/O pins when configuring the same configuration data into multiple devices.

You can use a single configuration chain to configure Cyclone II devices with other Altera devices. Connect all the Cyclone II device's and all other Altera device's CONF_DONE and nSTATUS pins together so all devices in the chain complete configuration at the same time or that an error reported by one device initiates reconfiguration in all devices.

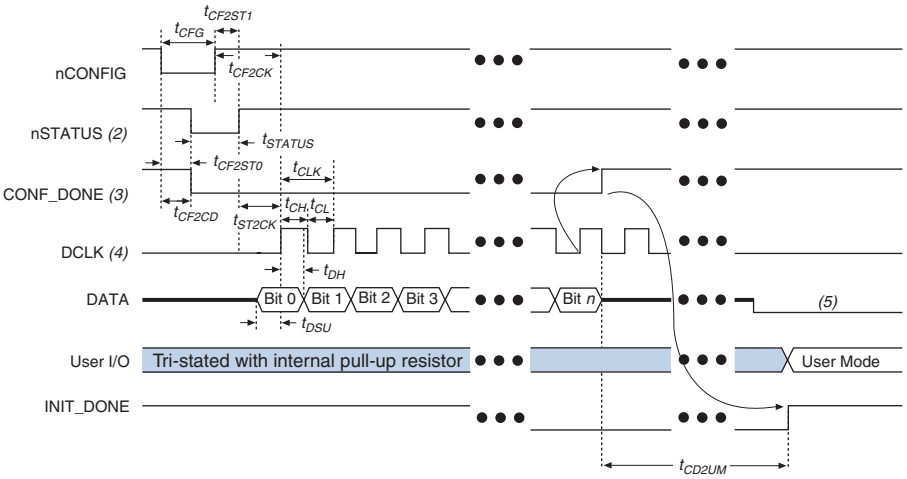
For more information on configuring multiple Altera devices in the same configuration chain, see *Configuring Mixed Altera FPGA Chains* in the *Configuration Handbook*.

PS Configuration Timing

A PS configuration must meet the setup and hold timing parameters and the maximum clock frequency. When using a microprocessor or another intelligent host to control the PS interface, ensure that you meet these timing requirements.

Figure 13–12 shows the timing waveform for PS configuration for Cyclone II devices.

Figure 13–12. PS Configuration Timing Waveform *Note (1)*



Notes to Figure 13–12:

(1) The beginning of this waveform shows the device in user mode. In user mode, nCONFIG, nSTATUS and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.
(2) Upon power-up, the Cyclone II device holds nSTATUS low for the time of the POR delay.
(3) Upon power-up, before and during configuration, CONF_DONE is low.
(4) In user mode, drive DCLK either high or low when using the PS configuration scheme, whichever is more convenient. When using the AS configuration scheme, DCLK is a Cyclone II output pin and should not be driven externally.
(5) Do not leave the DATA pin floating after configuration. Drive it high or low, whichever is more convenient.

Table 13–7 defines the timing parameters for Cyclone II devices for PS configuration.

Table 13–7. PS Timing Parameters for Cyclone II Devices

Symbol	Parameter	Minimum	Maximum	Units
t_{POR}	POR delay *(1)*	100		ms
t_{CF2CD}	nCONFIG low to CONF_DONE low		800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		800	ns
t_{CFG}	nCONFIG low pulse width	2		µs
t_{STATUS}	nSTATUS low pulse width	10	40 *(2)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high		40 *(2)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	40		µs
t_{ST2CK}	nSTATUS high to first rising edge on DCLK	1		µs
t_{DSU}	Data setup time before rising edge on DCLK	7		ns
t_{DH}	Data hold time after rising edge on DCLK	0		ns
t_{CH}	DCLK high time	4		ns
t_{CL}	DCLK low time	4		ns
t_{CLK}	DCLK period	10		ns
f_{MAX}	DCLK frequency		100	MHz
t_{CD2UM}	CONF_DONE high to user mode *(3)*	18	40	µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period		
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	t_{CD2CU} + (299 × CLKUSR period)		

Notes to Table 13–7:

(1) The POR delay minimum of 100 ms only applies for non "A" devices.

(2) This value is applicable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(3) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting the device.

Device configuration options and how to create configuration files are discussed further in the *Software Settings* section in Volume 2 of the *Configuration Handbook*.

PS Configuration Using a Microprocessor

In the PS configuration scheme, a microprocessor can control the transfer of configuration data from a storage device, such as flash memory, to the target Cyclone II device.

All information in the "Single Device PS Configuration Using a MAX II Device as an External Host" on page 13–22 section is also applicable when using a microprocessor as an external host. Refer to that section for all configuration information.

The MicroBlaster™ software driver allows you to configure Altera FPGAs, including Cyclone II devices, through the ByteBlaster II or ByteBlasterMV cable in PS mode. The MicroBlaster software driver supports a RBF programming input file and is targeted for embedded PS configuration. The source code is developed for the Windows NT operating system, although you can customize it to run on other operating systems.

Since the Cyclone II device can decompress the compressed configuration data on-the-fly during PS configuration, the MicroBlaster software can accept a compressed RBF file as its input file.

For more information on the MicroBlaster software driver, see the *Configuring the MicroBlaster Passive Serial Software Driver White Paper* and source files on the Altera web site at **www.altera.com**.

If you turn on the **Enable user-supplied start-up clock (CLKUSR)** option in the Quartus II software, the Cyclone II devices does not enter user mode after the MicroBlaster has transmitted all the configuration data in the RBF file. You need to supply enough initialization clock cycles to `CLKUSR` pin to enter user mode.

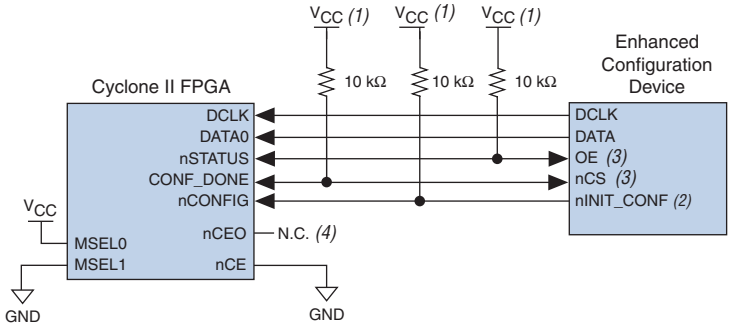
Single Device PS Configuration Using a Configuration Device

You can use an Altera configuration device (for example, an EPC2, EPC1, or enhanced configuration device) to configure Cyclone II devices using a serial configuration bitstream. Configuration data is stored in the configuration device. Figure 13–13 shows the configuration interface connections between the Cyclone II device and a configuration device.

The figures in this chapter only show the configuration-related pins and the configuration pin connections between the configuration device and the FPGA.

For more information on enhanced configuration devices and flash interface pins (e.g., `PGM[2..0]`, `EXCLK`, `PORSEL`, `A[20..0]`, and `DQ[15..0]`), see the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet*.

Figure 13–13. Single Device PS Configuration Using an Enhanced Configuration Device



Notes to Figure 13–13:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device. This pull-up resistor is 10 kΩ

(2) The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the nINIT_CONF to nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor (if reconfiguration is required, a resistor is necessary).

(3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

(4) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed other device's nCE pin.

The value of the internal pull-up resistors on the enhanced configuration devices and EPC2 devices can be found in the *Enhanced Configuration Devices (EPC4, EPC8, & EPC16) Data Sheet* or the *Configuration Devices for SRAM-based LUT Devices Data Sheet*.

When using enhanced configuration devices or EPC2 devices, you can connect the Cyclone II nCONFIG pin to the configuration device nINIT_CONF pin, which allows the INIT_CONF JTAG instruction to initiate FPGA configuration. You do not need to connect the nINIT_CONF pin if you are not using it. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), pull the nCONFIG signal to V_{CC} either directly or through a resistor (if reconfiguration is required, a resistor is necessary). An internal pull-up resistor on the nINIT_CONF pin is always active in enhanced configuration devices and EPC2 devices. Therefore, you do not need an external pull-up if nCONFIG is connected to nINIT_CONF.

Upon power-up, the Cyclone II device goes through a POR. During POR, the device reset, holds nSTATUS and CONF_DONE low, and tri-states all user I/O pins. After POR, which typically lasts 100 ms, the Cyclone II FPGA releases nSTATUS and enters configuration mode when this signal is pulled high by the external 10-kΩ resistor. Once the FPGA successfully exits POR, all user I/O pins continue to be tri-stated. Cyclone II devices have weak pull-up resistors on the user I/O pins which are on before and during configuration.

The configuration device also goes through a POR delay to allow the power supply to stabilize. The maximum POR time for EPC2 or EPC1 devices is 200 ms. The POR time for enhanced configuration devices can be set to 100 ms or 2 ms, depending on the enhanced configuration device's PORSEL pin setting. If the PORSEL pin is connected to ground, the POR delay is 100 ms. If the PORSEL pin is connected to V_{CC}, the POR delay is 2 ms. You must power the Cyclone II device before or during the enhanced configuration device POR time. During POR, the configuration device transitions its OE pin low. This low signal delays configuration because the OE pin is connected to the target device's nSTATUS pin. When the target and configuration devices complete POR, they both release the nSTATUS to OE line, which is then pulled high by a pull-up resistor.

When the power supplies have reached the appropriate operating voltages, the target FPGA senses the low-to-high transition on nCONFIG and initiates the configuration cycle. The configuration cycle consists of three stages: reset, configuration, and initialization.

The Cyclone II device does not have a PORSEL pin.

Reset Stage

While nCONFIG or nSTATUS is low, the device is in reset. You can delay configuration by holding the nCONFIG or nSTATUS pin low.

V_{CCINT} and V_{CCIO} of the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When the nCONFIG signal goes high, the device comes out of reset and releases the nSTATUS pin, which is pulled high by a pull-up resistor. Enhanced configuration and EPC2 devices have an optional internal pull-up resistor on the OE pin. You can turn on this option in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If this internal pull-up resistor is not used, you need to connect an external 10-kΩ pull-up resistor to the OE and nSTATUS line. Once nSTATUS is released, the FPGA is ready to receive configuration data and the configuration stage begins.

Configuration Stage

When the nSTATUS pin transitions high, the configuration device's OE pin also transitions high and the configuration device clocks data out serially to the FPGA using its internal oscillator. The Cyclone II device receives configuration data on its DATA0 pin and the clock is received on the DCLK pin. Data is latched into the FPGA on the rising edge of DCLK.

After the FPGA has received all configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by a pull-up resistor. Since the Cyclone II device's CONF_DONE pin is tied to the configuration device's nCS pin, the configuration device is disabled when CONF_DONE goes high. Enhanced configuration and EPC2 devices have an optional internal pull-up resistor on the nCS pin. You can turn this option on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If you do not use this internal pull-up resistor, you need to connect an external 10-kΩ pull-up resistor to the nCS and CONF_DONE line. A low-to-high transition on CONF_DONE indicates configuration is complete, and the device can begin initialization.

Initialization Stage

In Cyclone II devices, the default initialization clock source is the Cyclone II internal oscillator (typically 10 MHz). Cyclone II devices can also use the optional CLKUSR pin. If your design uses the internal oscillator, the Cyclone II device supplies itself with enough clock cycles for proper initialization. The advantage of using the internal oscillator is you do not need to use another device or source to send additional clock cycles to the CLKUSR pin during the initialization stage. Additionally, you can use of the CLKUSR pin as a user I/O pin, which means you have an additional user I/O pin.

If you want to delay the initialization of the device, you can use the CLKUSR pin. Using the CLKUSR pin allows you to control when the Cyclone II device enters user mode. You can delay the Cyclone II devices from entering user mode for an indefinite amount of time. You can turn on the **Enable user-supplied start-up clock (CLKUSR)** option in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR does not affect the configuration process. After all configuration data is accepted and CONF_DONE goes high, Cyclone II devices require 299 clock cycles to properly initialize and support a CLKUSR f_{MAX} of 100 MHz.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user mode with a low-to-high transition. The **Enable INIT_DONE output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If you use the INIT_DONE pin, an external 10-kΩ pull-up resistor pulls it high when

nCONFIG is low and during the beginning of configuration. Once the optional bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. This low-to-high transition signals that the FPGA has entered user mode. If you do not use the INIT_DONE pin, the initialization period is complete after the CONF_DONE signal transitions high and 299 clock cycles are sent to the CLKUSR pin or after the time t_{CF2UM} (see Table 13–7) if the Cyclone II device uses the internal oscillator.

After successful configuration, if you intend to synchronize the initialization of multiple devices that are not in the same configuration chain, your system must not pull the CONF_DONE signal low to delay initialization. Instead, use the optional CLKUSR pin to synchronize the initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain initialize together if their CONF_DONE pins are tied together.

If the optional CLKUSR pin is being used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure that CLKUSR continues toggling during the time nSTATUS is low (maximum of 40 µs).

User Mode

When initialization is complete, the FPGA enters user mode. In user mode, the user I/O pins do not have weak pull-up resistors and function as assigned in your design. Enhanced configuration devices and EPC2 devices drive DCLK low and DATA0 high (EPC1 devices drive the DCLK pin low and tri-state the DATA pin) at the end of configuration.

When the FPGA is in user mode, pull the nCONFIG pin low to begin reconfiguration. The nCONFIG pin should be low for at least 2 µs. When nCONFIG transitions low, the Cyclone II device also pulls the nSTATUS and CONF_DONE pins low and all I/O pins are tri-stated. Because CONF_DONE transitions low, this activates the configuration device since it will see its nCS pin transition low. Once nCONFIG returns to a logic high level and nSTATUS is released by the FPGA, reconfiguration begins.

Error During Configuration

If an error occurs during configuration, the Cyclone II drives its nSTATUS pin low, resetting itself internally. Since the nSTATUS pin is tied to OE, the configuration device is also reset. If you turn on the **Auto-restart configuration after error** option in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box, the FPGA automatically initiates reconfiguration if an error occurs. The Cyclone II

device releases its nSTATUS pin after a reset time-out period (maximum of 40 µs). When the nSTATUS pin is released and pulled high by a pull-up resistor, the configuration device reconfigures the chain. If this option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 2 µs to restart configuration. The external system can pulse the nCONFIG pin if the pin is under system control rather than tied to V_{CC}.

Additionally, if the configuration device sends all of its data and then detects that the CONF_DONE pin has not transitioned high, it recognizes that the FPGA has not configured successfully. Enhanced configuration devices wait for 64 DCLK cycles after the last configuration bit was sent for the CONF_DONE pin to transition high. EPC2 devices wait for 16 DCLK cycles. After that, the configuration device pulls its OE pin low, which in turn drives the target device's nSTATUS pin low. If you turn on the **Auto-restart configuration after error** option in the Quartus II software, the target device resets and then releases its nSTATUS pin after a reset time-out period (maximum of 40 µs). When nSTATUS transitions high again, the configuration device reconfigures the FPGA.

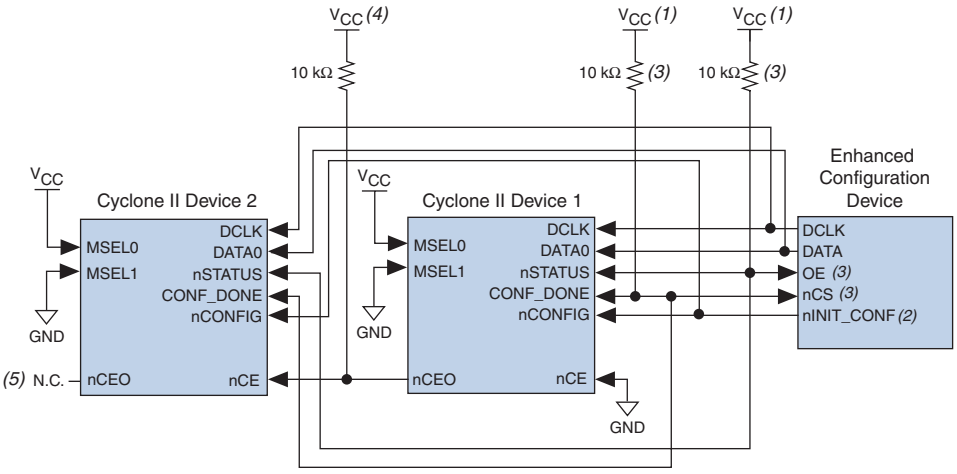
For more information on configuration issues, see the *Debugging Configuration Problems* chapter of the *Configuration Handbook* and the FPGA Configuration Troubleshooter on the Altera web site (**www.altera.com**).

Multiple Device PS Configuration Using a Configuration Device

You can use Altera enhanced configuration devices (EPC16, EPC8, and EPC4 devices) or EPC2 and EPC1 configuration devices to configure multiple Cyclone II devices in a PS configuration chain.

Figure 13–14 shows how to configure multiple devices with an enhanced configuration device. This circuit is similar to the configuration device circuit for a single device, except Cyclone II devices are cascaded for multiple device configuration.

Figure 13–14. Multiple Device PS Configuration Using an Enhanced Configuration Device



Notes to Figure 13–14:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the nINIT_CONF to nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor (if reconfiguration is required, a resistor is necessary).

(3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

(4) Connect the pull-up resistor to the V_{CCIO} supply voltage of I/O bank that the nCEO pin resides in.

(5) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed other device's nCE pin.

You cannot cascade enhanced configuration devices (EPC16, EPC8, and EPC4 devices).

When configuring multiple devices, you must generate the configuration device's POF from each project's SOF. You can combine multiple SOFs using the **Convert Programming Files** window in the Quartus II software.

For more information on how to create configuration files for multiple device configuration chains, see the *Software Settings* section in Volume 2 of the *Configuration Handbook*.

When configuring multiple devices with the PS scheme, connect the first Cyclone II device's nCE pin to GND and connect its nCEO pin to the nCE pin of the Cyclone II device in the chain. Use an external 10-kΩ pull-up resistor to pull the Cyclone II device's nCEO pin to the V_{CCIO} level when

it feeds the next device's nCE pin. After the first device in the chain completes configuration, its nCEO pin transitions low to activate the second device's nCE pin, which prompts the second device to begin configuration. You can leave the nCEO pin of the last device unconnected or use it as a user I/O pin after configuration. The nCEO pin is a dual-purpose pin in Cyclone II devices.

The Quartus II software sets the Cyclone II device nCEO pin as an output pin driving to ground by default. If the device is in a chain, and the nCEO pin is connected to the next device's nCE pin, you must make sure that the nCEO pin is not used as a user I/O pin after configuration. This software setting is in the **Dual-Purpose Pins** tab of the **Device & Pin Options** dialog box in Quartus II software.

Connect all other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) to every Cyclone II device in the chain. The configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Buffer the DCLK and DATA lines for every fourth device.

When configuring multiple devices, configuration does not begin until all devices release their OE or nSTATUS pins. Similarly, since all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

You should not pull CONF_DONE low to delay initialization. Instead, use the Quartus II software's **User-Supplied Start-Up Clock** option to synchronize the initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain initialize together since their CONF_DONE pins are tied together.

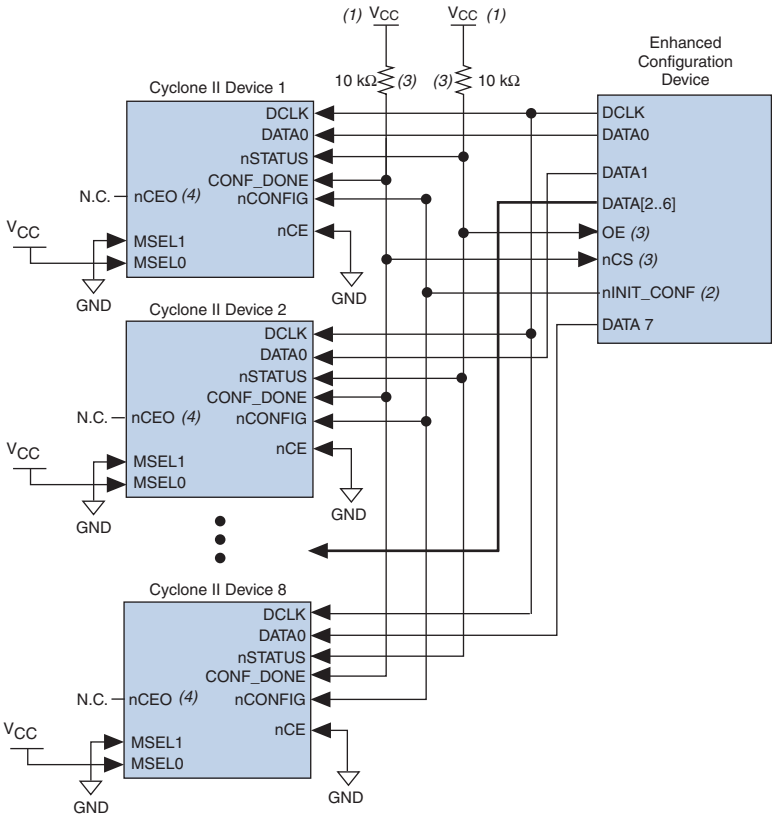
Since all nSTATUS and CONF_DONE pins are connected, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if there is an error when configuring the first Cyclone II device, it resets the chain by pulling its nSTATUS pin low. This low signal drives the OE pin low on the enhanced configuration device and drives nSTATUS low on all FPGAs, which causes them to enter a reset state.

If the **Auto-restart configuration after error** option is turned on, the devices automatically initiate reconfiguration if an error occurs. The FPGAs release their nSTATUS pins after a reset time-out period (40 µs maximum). When all the nSTATUS pins are released and pulled high, the configuration device reconfigures the chain. If the **Auto-restart configuration after error** option is turned off, a microprocessor or controller must monitor the nSTATUS pin for errors and then pulse

nCONFIG low for at least 2 µs to restart configuration. The microprocessor or controller can only transition the nCONFIG pin low if the pin is under system control and not tied to V_{CC}.

The enhanced configuration devices support parallel configuration of up to eight devices. The *n*-bit (*n* = 1, 2, 4, or 8) PS configuration mode allows enhanced configuration devices to concurrently configure a chain of FPGAs. These devices do not have to be the same device family or density; they can be any combination of Altera FPGAs with different designs. An individual enhanced configuration device DATA pin is available for each targeted FPGA. Each DATA line can also feed a chain of FPGAs. Figure 13–15 shows how to concurrently configure multiple devices using an enhanced configuration device.

Figure 13–15. Concurrent PS Configuration of Multiple Devices Using an Enhanced Configuration Device



Notes to Table 13–15:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the nINIT_CONF to nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor (if reconfiguration is required, a resistor is necessary).

(3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

(4) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed other device's nCE pin.

The Quartus II software only allows you to set *n* to 1, 2, 4, or 8. However, you can use these modes to configure any number of devices from 1 to 8. For example, if you configure three FPGAs, you would use the 4-bit PS mode. For the DATA0, DATA1, and DATA2 lines, the corresponding SOF data is transmitted from the configuration device to the FPGA. For

DATA3, you can leave the corresponding bit 3 line blank in the Quartus II software. On the printed circuit board (PCB), leave the DATA3 line from the enhanced configuration device unconnected. Use the Quartus II **Convert Programming Files** window (Tools menu) setup for this scheme.

You can also connect two FPGAs to one of the configuration device's DATA pins while the other DATA pins drive one device each. For example, you could use the 2-bit PS mode to drive two FPGAs with DATA bit 0 (two EP2C5 devices) and the third device (an EP2C8 device) with DATA bit 1. In this example, the memory space required for DATA bit 0 is the sum of the SOF file size for the two EP2C5 devices.

1,223,980 bits + 1,223,980 bits = 2,447,960 bits

The memory space required for DATA bit 1 is the SOF file size for on EP2C8 device (1,983,792 bits). Since the memory space required for DATA bit 0 is larger than the memory space required for DATA bit 1, the size of the POF file is 2 × 2,447,960 = 4,895,920.

For more information on using *n*-bit PS modes with enhanced configuration devices, see the *Using Altera Enhanced Configuration Devices* in the *Configuration Handbook*.

When configuring SRAM-based devices using *n*-bit PS modes, use Table 13–8 to select the appropriate configuration mode for the fastest configuration times.

Table 13–8. Recommended Configuration Using n-Bit PS Modes

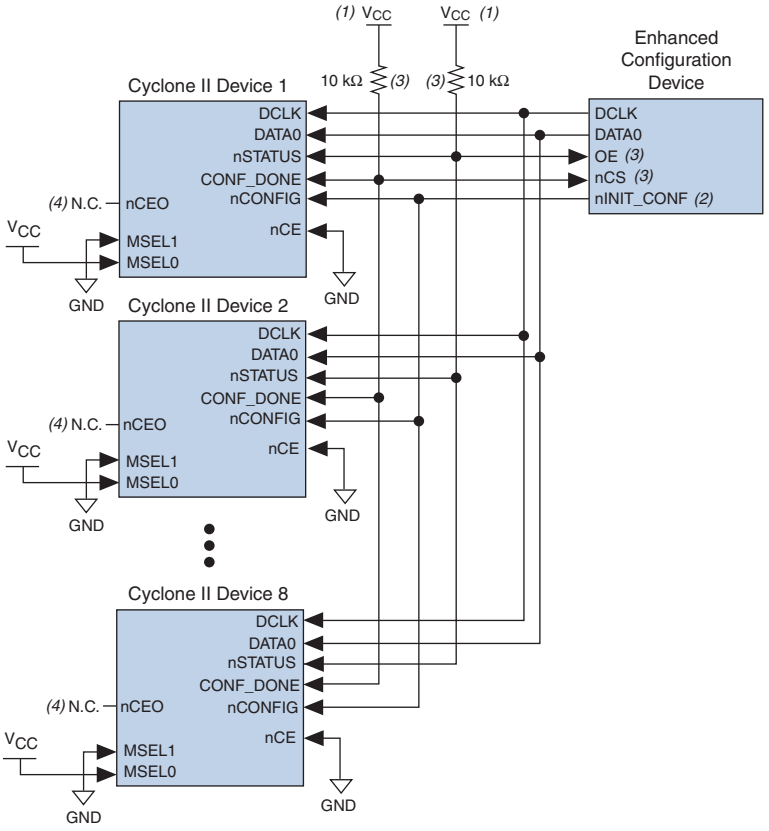
Number of Devices *(1)*	Recommended Configuration Mode
1	1-bit PS
2	2-bit PS
3	4-bit PS
4	4-bit PS
5	8-bit PS
6	8-bit PS
7	8-bit PS
8	8-bit PS

Note to Table 13–8:

(1) Assume that each DATA line is only configuring one device, not a daisy chain of devices.

If your design has multiple Cyclone II devices of the same density and package that contain the same configuration data, connect the nCE inputs to GND and leave the nCEO pins floating. You can also use the nCEO pin as a user I/O pin. Connect the configuration device nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE pins to each Cyclone II device in the chain. The configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Make sure that the DCLK and DATA lines are buffered for every fourth device. All devices start and complete configuration at the same time. Figure 13–16 shows multiple device PS configuration when the Cyclone II devices are receiving the same configuration data.

Figure 13–16. Multiple Device PS Configuration Using an Enhanced Configuration Device When FPGAs Receive the Same Data



Notes to Figure 13–16:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the nINIT_CONF to nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor (if reconfiguration is required, a resistor is necessary).

(3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

(4) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed other device's nCE pin.

You can cascade several EPC2 or EPC1 devices to configure multiple Cyclone II devices. The first configuration device in the chain is the master configuration device, and the subsequent devices are the slave devices. The master configuration device sends DCLK to the Cyclone II

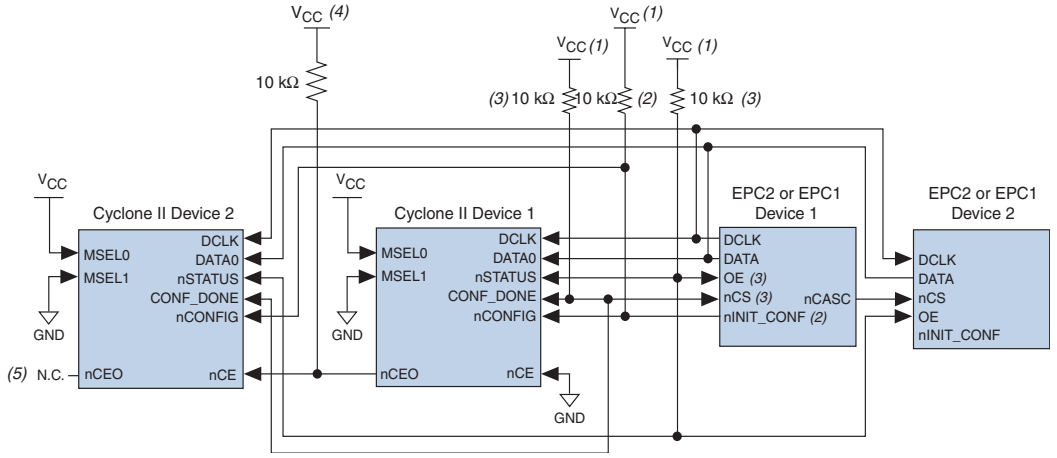
devices and to the slave configuration devices. Connect the first configuration device's nCS pin to all the Cyclone II device's CONF_DONE pins, and connect the nCASC pin to the nCS pin of the next configuration device in the chain. Leave the nCASC pin of the last configuration device floating. When the master configuration device sends all the data to the Cyclone II device, the configuration device transitions the nCASC pin low, which drives nCS on the next configuration device. Because a configuration device requires less than one clock cycle to activate a subsequent configuration device, the data stream is uninterrupted.

Enhanced configuration devices (EPC16, EPC8, and EPC4 devices) cannot be cascaded.

Since all nSTATUS and CONF_DONE pins are connected, if any device detects an error, the master configuration device stops configuration for the entire chain and the entire chain must be reconfigured. For example, if the master configuration device does not detect the Cyclone II device's CONF_DONE pin transitioning high at the end of configuration, it resets the entire chain by transitioning its OE pin low. This low signal drives the OE pin low on the slave configuration device(s) and drives nSTATUS low on all Cyclone II devices, causing them to enter a reset state. This behavior is similar to the FPGA detecting an error in the configuration data.

Figure 13–17 shows how to configure multiple devices using cascaded EPC2 or EPC1 devices.

Figure 13–17. Multiple Device PS Configuration Using Cascaded EPC2 or EPC1 Devices



Notes to Figure 13–17:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the nINIT_CONF to nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CC} either directly or through a resistor (if reconfiguration is required, a resistor is necessary).

(3) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device option** when generating programming files.

(4) Use an external 10-kΩ pull-up resistor to pull the nCEO pin high to the I/O bank V_{CCIO} level to help the internal weak pull-up when it feeds next device's nCE pin.

(5) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed other device's nCE pin.

When using enhanced configuration devices or EPC2 devices, you can connect the Cyclone II device's nCONFIG pin to the configuration device's nINIT_CONF pin, which allows the INIT_CONF JTAG instruction to initiate FPGA configuration. You do not need to connect the nINIT_CONF pin if it is not used. If the nINIT_CONF pin is not used or not available (for example, on EPC1 devices), pull the nCONFIG pin to V_{CC} levels either directly or through a resistor (if reconfiguration is required, a resistor is necessary). An internal pull-up resistor on the nINIT_CONF pin is always active in the enhanced configuration devices and the EPC2 devices. Therefore, do not use an external pull-up resistor if you connect the nCONFIG pin to nINIT_CONF. If you use multiple EPC2 devices to configure a Cyclone II device(s), only connect the first EPC2 device's nINIT_CONF pin to the device's nCONFIG pin.

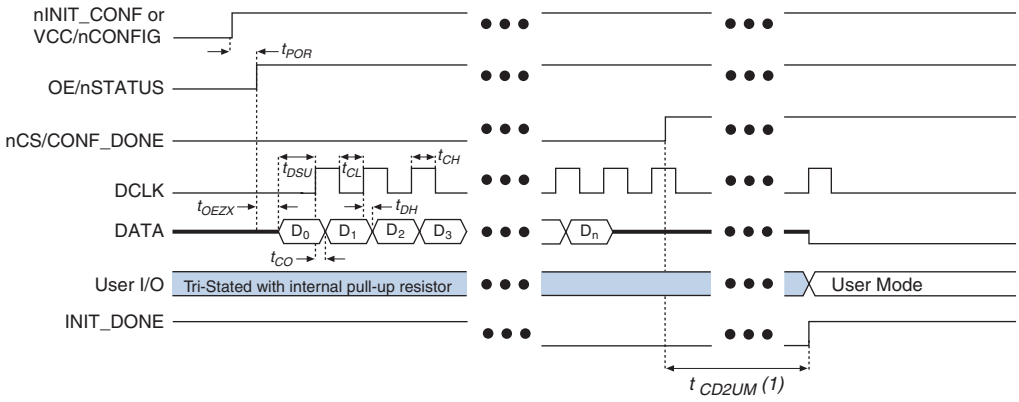
You can use a single configuration chain to configure Cyclone II devices with other Altera devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, connect all the Cyclone II device CONF_DONE pins and connect all Cyclone II device nSTATUS pins together.

For more information on configuring multiple Altera devices in the same configuration chain, see the *Configuring Mixed Altera FPGA Chains* chapter in the *Configuration Handbook*.

During PS configuration, the design must meet the setup and hold timing parameters and maximum DCLK frequency. The enhanced configuration and EPC2 devices are designed to meet these interface timing specifications.

Figure 13–18 shows the timing waveform for the PS configuration scheme using a configuration device.

Figure 13–18. Cyclone II PS Configuration Using a Configuration Device Timing Waveform



Note to Figure 13–18:

(1) Cyclone II devices enter user mode 299 clock cycles after CONF_DONE goes high. The initialization clock can come from the Cyclone II internal oscillator or the CLKUSR pin.

For timing information, refer to the *Enhanced Configuration Devices (EPC4, EPC8, and EPC16) Data Sheet* or the *Configuration Devices for SRAM-based LUT Devices Data Sheet* in the *Configuration Handbook*.

For more information on device configuration options and how to create configuration files, see the *Software Settings* section in Volume 2 of the *Configuration Handbook*.

PS Configuration Using a Download Cable

In PS configuration, an intelligent host (e.g., a PC) can use a download cable to transfer data from a storage device to the Cyclone II device. You can use the Altera USB-Blaster universal serial bus (USB) port download cable, MasterBlaster™ serial/USB communications cable, ByteBlaster II parallel port download cable, or the ByteBlasterMV™ parallel port as a download cable.

Upon power up, the Cyclone II device goes through POR, which lasts approximately 100 ms for non "A" devices. During POR, the device resets, holds nSTATUS low, and tri-states all user I/O pins. Once the FPGA successfully exits POR, the nSTATUS pin is released and all user I/O pins continue to be tri-stated.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *Cyclone II Device Handbook*.

The configuration cycle consists of three stages: reset, configuration, and initialization. While the nCONFIG or nSTATUS pins are low, the device is in reset. To initiate configuration in this scheme, the download cable generates a low-to-high transition on the nCONFIG pin.

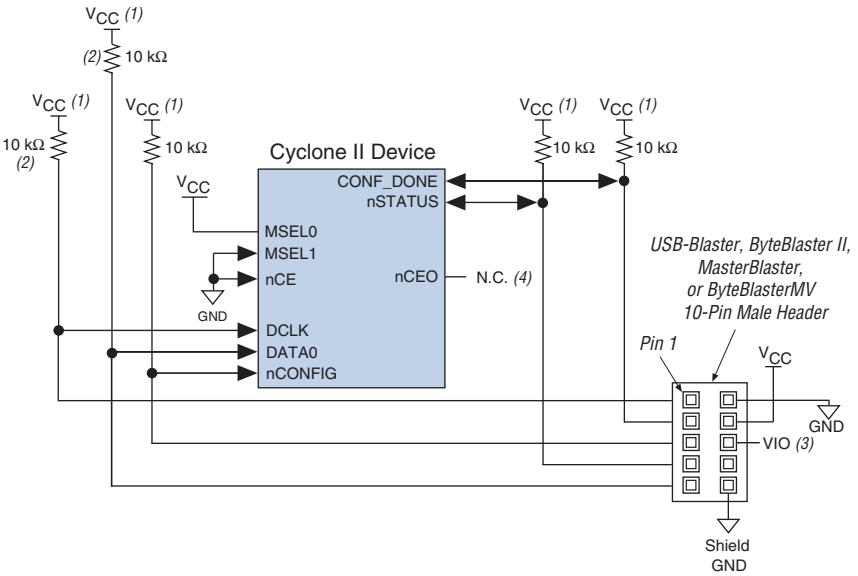
Make sure V_{CCINT} and V_{CCIO} for the banks where the configuration and JTAG pins reside are powered to the appropriate voltage levels in order to begin the configuration process.

When nCONFIG transitions high, the Cyclone II device comes out of reset and begins configuration. The Cyclone II device releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. Once nSTATUS transitions high, the Cyclone II device is ready to receive configuration data. The programming hardware or download cable then transmits the configuration data one bit at a time to the device's DATA0 pin. The configuration data is clocked into the target device until CONF_DONE goes high. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

When using a download cable, you cannot use the **Auto-restart configuration after error** option. You must manually restart configuration in the Quartus II software when an error occurs. Additionally, you cannot use the **Enable user-supplied start-up clock (CLKUSR)** option when programming the FPGA using the Quartus II programmer and download cable. This option is disabled in the SOF. Therefore, if you turn on the CLKUSR option, you do not need to provide a clock on CLKUSR when you are configuring the FPGA with the

Quartus II programmer and a download cable. Figure 13–19 shows the PS configuration for Cyclone II devices using a USB-Blaster, MasterBlaster, ByteBlaster II or ByteBlasterMV cable.

Figure 13–19. PS Configuration Using a USB-Blaster, MasterBlaster, ByteBlaster II or ByteBlasterMV Cable



Notes to Figure 13–19:

(1) The pull-up resistor should be connected to the same supply voltage as the USB-Blaster, MasterBlaster (VIO pin), ByteBlaster II, or ByteBlasterMV cable.

(2) The pull-up resistors on DATA0 and DCLK are only needed if the download cable is the only configuration scheme used on your board. This is to ensure that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, the pull-up resistors on DATA0 and DCLK are not needed.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. Refer to the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value. In the ByteBlasterMV, this pin is a no connect. In the USB-Blaster and ByteBlaster II, this pin is connected to nCE when it is used for AS programming, otherwise it is a no connect.

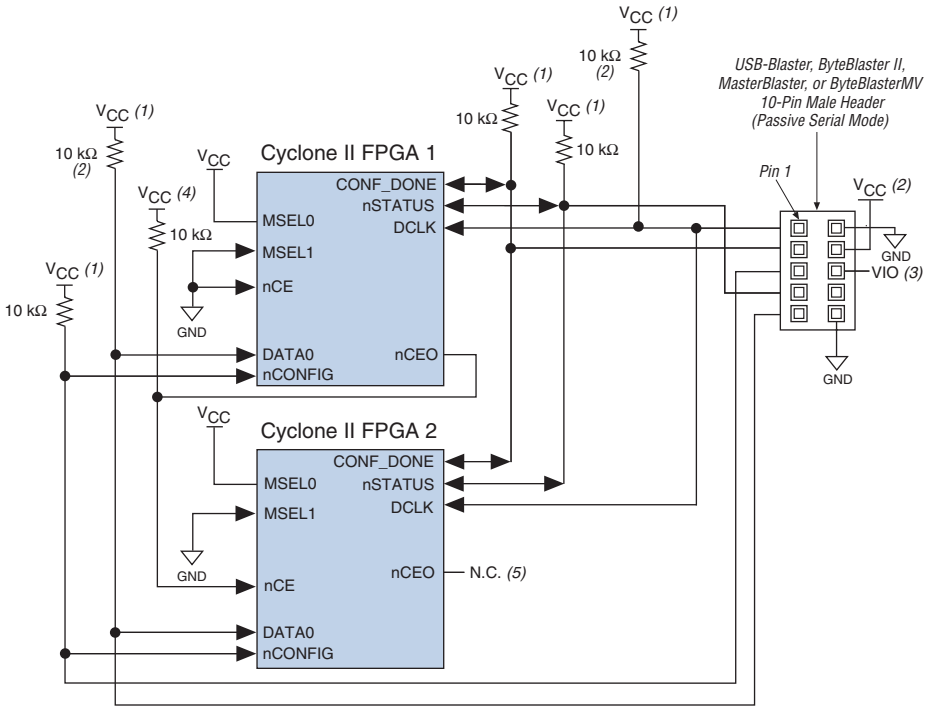
(4) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed other device's nCE pin.

You can use a download cable to configure multiple Cyclone II devices by connecting each device's nCEO pin to the subsequent device's nCE pin. Connect the first Cyclone II device's nCE pin to GND and connect its nCEO pin to the nCE pin of the next device in the chain. Use an external 10-kΩ pull-up resistor to pull the nCEO pin high to V_{CCIO} when it feeds next device's nCE pin. Connect all other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) on every device in the chain together. Because all CONF_DONE pins are connected, all devices in the chain initialize and enter user mode at the same time.

In addition, because the nSTATUS pins are connected, all the Cyclone II devices in the chain stop configuration if any device detects an error. If this happens, you must manually restart configuration in the Quartus II software.

Figure 13–20 shows how to configure multiple Cyclone II devices with a download cable.

Figure 13–20. Multiple Device PS Configuration Using a USB-Blaster, MasterBlaster, ByteBlaster II or ByteBlasterMV Cable



Notes to Figure 13–20:

(1) The pull-up resistor should be connected to the same supply voltage as the USB-Blaster, MasterBlaster (VIO pin), ByteBlaster II, or ByteBlasterMV cable.

(2) The pull-up resistors on DATA0 and DCLK are only needed if the download cable is the only configuration scheme used on your board. This is to ensure that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, the pull-up resistors on DATA0 and DCLK are not needed.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. Refer to the MasterBlaster Serial/USB Communications Cable Data Sheet for this value. In the ByteBlasterMV, this pin is a no connect. In the USB-Blaster and ByteBlaster II, this pin is connected to nCE when it is used for AS programming, otherwise it is a no connect.

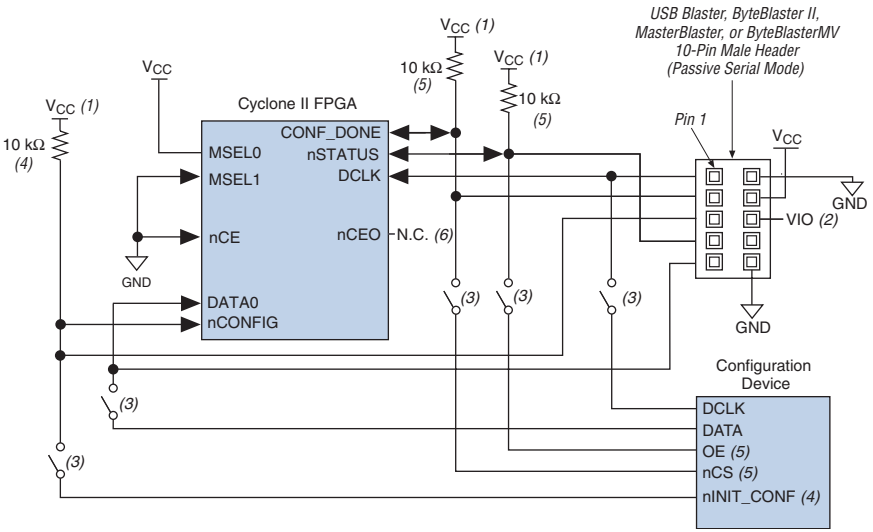
(4) Connect the pull-up resistor to the V_{CCIO} supply voltage of I/O bank that the nCEO pin resides in.

(5) The nCEO pin of the last device in chain can be left unconnected or used as a user I/O pin.

If you are using a download cable to configure Cyclone II devices on a PCB that also has configuration devices, you should electrically isolate the configuration devices from the target Cyclone II devices and cable. One way to isolate the configuration device is to add logic, such as a multiplexer, that can select between the configuration device and the cable. The multiplexer should allow bidirectional transfers on the nSTATUS and CONF_DONE signals. Additionally, you can add switches to

the five common signals (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) between the cable and the configuration device. You can also remove the configuration device from the board when configuring the FPGA with the cable. Figure 13–21 shows a combination of a configuration device and a download cable to configure an FPGA.

Figure 13–21. PS Configuration with a Download Cable & Configuration Device Circuit



Notes to Figure 13–21:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. Refer to the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value. In the ByteBlasterMV, this pin is a no connect. In the USB-Blaster and ByteBlaster II, this pin is connected to nCE when it is used for AS programming, otherwise it is a no connect.

(3) You should not attempt configuration with a download cable while a configuration device is connected to a Cyclone II device. Instead, you should either remove the configuration device from its socket when using the download cable or place a switch on the five common signals between the download cable and the configuration device.

(4) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) has an internal pull-up resistor that is always active. This means an external pull-up resistor should not be used on the nINIT_CONF to nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CC} either directly or through a resistor (if reconfiguration is required, a resistor is necessary).

(5) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

(6) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed other device's nCE pin.

For more information on how to use the USB-Blaster, MasterBlaster, ByteBlaster II or ByteBlasterMV cables, refer to the following documents:

- *USB-Blaster USB Port Download Cable Data Sheet*
- *MasterBlaster Serial/USB Communications Cable Data Sheet*
- *ByteBlaster II Parallel Port Download Cable Data Sheet*
- *ByteBlasterMV Parallel Port Download Cable Data Sheet*

JTAG Configuration

The Joint Test Action Group (JTAG) has developed a specification for boundary-scan testing. This boundary-scan test (BST) architecture allows you to test components on PCBs with tight lead spacing. The BST architecture can test pin connections without using physical test probes and capture functional data while a device is operating normally. The JTAG circuitry can also be used to shift configuration data into the device. The Quartus II software automatically generates SOF files that can be used for JTAG configuration with a download cable in the Quartus II programmer.

For more information on JTAG boundary-scan testing, see the following documents:

- *IEEE 1149.1 (JTAG) Boundary-Scan Testing for Cyclone II Devices* chapter in Volume 2 of the *Cyclone II Device Handbook*
- Jam Programming & Testing Language Specification

Cyclone II devices are designed such that JTAG instructions have precedence over any device configuration modes. This means that JTAG configuration can take place without waiting for other configuration modes to complete. For example, if you attempt JTAG configuration of Cyclone II devices during PS configuration, PS configuration terminates and JTAG configuration begins. If the Cyclone II `MSEL` pins are set to AS or fast AS mode, the Cyclone II device does not output a `DCLK` signal when JTAG configuration takes place.

You cannot use the Cyclone II decompression feature if you are configuring your Cyclone II device when using JTAG-based configuration.

A device operating in JTAG mode uses the TDI, TDO, TMS, and TCK pins. The TCK pin has a weak internal pull-down resistor while the other JTAG input pins, TDI and TMS, have weak internal pull-up resistors. All user I/O pins are tri-stated during JTAG configuration. Table 13–9 explains each JTAG pin's function.

Table 13–9. Dedicated JTAG Pins

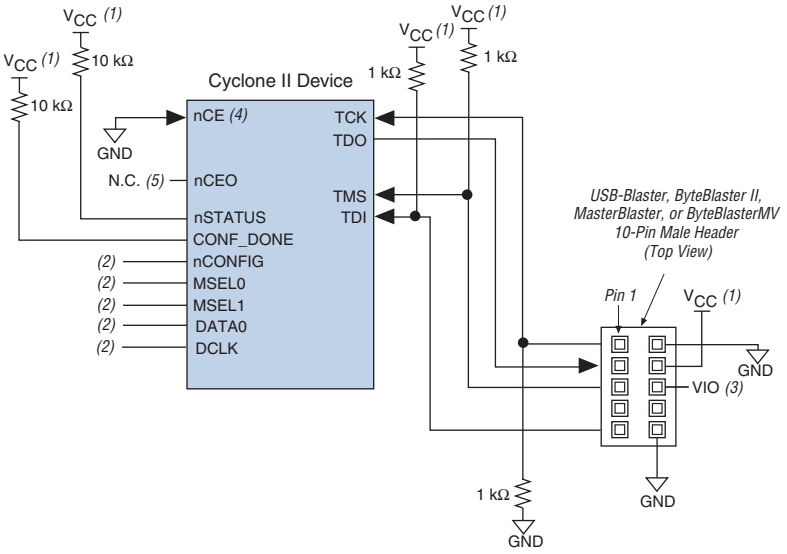
Pin Name	Pin Type	Description
TDI	Test data input	Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}.
TDO	Test data output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by leaving this pin unconnected.
TMS	Test mode select	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}.
TCK	Test clock input	The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND.

The TDO output is powered by the V_{CCIO} power supply. If V_{CCIO} is tied to 3.3-V, both the I/O pins and the JTAG TDO port drive at 3.3-V levels.

Single Device JTAG Configuration

During JTAG configuration, you can use the USB-Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV download cable to download data to the device. Configuring Cyclone II devices through a cable is similar to programming devices in system. Figure 13–22 shows JTAG configuration of a single Cyclone II device using a download cable.

Figure 13–22. JTAG Configuration of a Single Device Using a Download Cable



Notes to Figure 13–22:

(1) The pull-up resistor should be connected to the same supply voltage as the USB-Blaster, MasterBlaster (VIO pin), ByteBlaster II, or ByteBlasterMV cable.

(2) Connect the nCONFIG and MSEL[1..0] pins to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect the nCONFIG pin to V_{CC}, and the MSEL[1..0] pins to ground. In addition, pull DCLK and DATA0 to either high or low, whichever is convenient on your board.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. Refer to the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value. In the ByteBlasterMV, this pin is a no connect. In the USB-Blaster and ByteBlaster II, this pin is connected to nCE when it is used for AS programming, otherwise it is a no connect.

(4) nCE must be connected to GND or driven low for successful JTAG configuration.

(5) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed other device's nCE pin.

To configure a single device in a JTAG chain, the programming software places all other devices in BYPASS mode. In BYPASS mode, Cyclone II devices pass programming data from the TDI pin to the TDO pin through a single bypass register without being affected internally. This scheme

enables the programming software to program or verify the target device. Configuration data driven into the target device appears on the TDO pin one clock cycle later.

The Quartus II software verifies successful JTAG configuration upon completion. At the end of configuration, the software checks the CONF_DONE pin through the JTAG port. When the Quartus II software generates a JAM file for a multiple device chain, it contains instructions so that all the devices in the chain are initialized at the same time. If CONF_DONE is not high, the Quartus II software indicates that configuration has failed. If the CONF_DONE pin transitions high, the software indicates that configuration was successful. After the configuration bitstream is transmitted serially via the JTAG TDI port, the TCK port is clocked an additional 299 cycles to perform Cyclone II device initialization.

The **Enable user-supplied start-up clock (CLKUSR)** option has no affect on the device initialization since this option is disabled in the SOF when configuring the FPGA in JTAG using the Quartus II programmer and download cable. Therefore, if you turn on the CLKUSR option, you do not need to provide a clock on CLKUSR when you are configuring the FPGA with the Quartus II programmer and a download cable.

Cyclone II devices have dedicated JTAG pins that always function as JTAG pins. You can perform JTAG testing on Cyclone II devices before, after, and during configuration. Cyclone II devices support the BYPASS, IDCODE and SAMPLE instructions during configuration without interruption. All other JTAG instructions may only be issued by first interrupting configuration and reprogramming I/O pins using the CONFIG_IO instruction.

The CONFIG_IO instruction allows I/O buffers to be configured via the JTAG port. The CONFIG_IO instruction interrupts configuration. This instruction allows you to perform board-level testing before configuring the Cyclone II device or waiting for a configuration device to complete configuration. If you interrupt configuration, the Cyclone II device must be reconfigured via JTAG (PULSE_CONFIG instruction) or by pulsing nCONFIG low after JTAG testing is complete.

For more information, see the *MorphIO: An I/O Reconfiguration Solution for Altera White Paper*.

The chip-wide reset (DEV_CLRn) and chip-wide output enable (DEV_OE) pins on Cyclone II devices do not affect JTAG boundary-scan or programming operations. Toggling these pins does not affect JTAG operations (other than the usual boundary-scan operation).

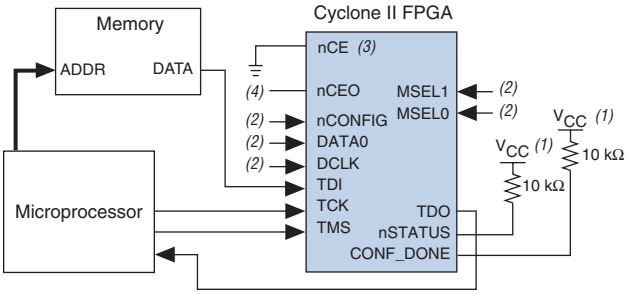
When designing a Cyclone II board for JTAG configuration, use the guidelines in Table 13–10 for the placement of the dedicated configuration pins.

Table 13–10. Dedicated Configuration Pin Connections During JTAG Configuration

Signal	Description
nCE	On all Cyclone II devices in the chain, nCE should be driven low by connecting it to ground, pulling it low via a resistor, or driving it by some control circuitry. For devices that are also in multiple device AS, or PS configuration chains, the nCE pins should be connected to GND during JTAG configuration or JTAG configured in the same order as the configuration chain.
nCEO	On all Cyclone II devices in the chain, nCEO can be used as a user I/O or connected to the nCE of the next device. If nCEO is connected to the nCE of the next device, the nCEO pin must be pulled high to V_{CCIO} by an external 10-kΩ pull-up resistor to help the internal weak pull-up resistor. If the nCEO pin is not connected to the nCE pin of the next device, you can use it as a user I/O pin after configuration.
MSEL	These pins must not be left floating. These pins support whichever non-JTAG configuration is used in production. If only JTAG configuration is used, you should tie these pins to ground.
nCONFIG	Driven high by connecting to V_{CC}, pulling up via a resistor, or driven high by some control circuitry.
nSTATUS	Pull to V_{CC} via a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each nSTATUS pin should be pulled up to V_{CC} individually. nSTATUS pulling low in the middle of JTAG configuration indicates that an error has occurred.
CONF_DONE	Pull to V_{CC} via a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each CONF_DONE pin should be pulled up to V_{CC} individually. CONF_DONE going high at the end of JTAG configuration indicates successful configuration.
DCLK	Should not be left floating. Drive low or high, whichever is more convenient on your board.

Figure 13–23 shows JTAG configuration of a Cyclone II device with a microprocessor.

Figure 13–23. JTAG Configuration of a Single Device Using a Microprocessor



Notes to Figure 13–23:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.

(2) Connect the nCONFIG and MSEL[1..0] pins to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect the nCONFIG pin to V_{CC}, and the MSEL[1..0] pins to ground. In addition, pull DCLK and DATA0 to either high or low, whichever is convenient on your board.

(3) nCE must be connected to GND or driven low for successful JTAG configuration.

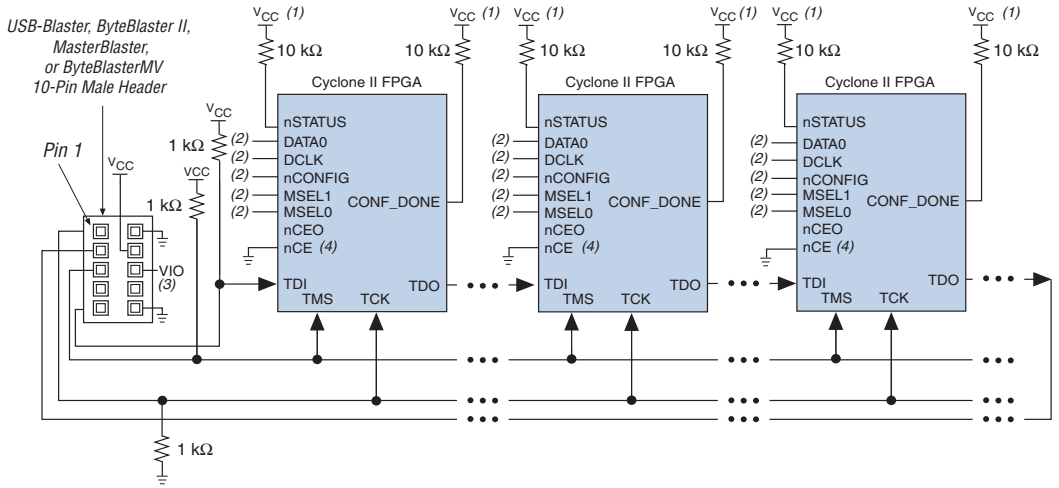
(4) If using an EPCS4 or EPCS1 device, set MSEL[1..0] to 00. See Table 13–4 for more details.

JTAG Configuration of Multiple Devices

When programming a JTAG device chain, one JTAG-compatible header is connected to several devices. The number of devices in the JTAG chain is limited only by the drive capability of the download cable. When four or more devices are connected in a JTAG chain, Altera recommends buffering the TCK, TDI, and TMS pins with an on-board buffer.

JTAG-chain device programming is ideal when the system contains multiple devices, or when testing your system using JTAG BST circuitry. Figure 13–24 shows multiple device JTAG configuration.

Figure 13–24. JTAG Configuration of Multiple Devices Using a Download Cable



Notes to Figure 13–24:

(1) The pull-up resistor should be connected to the same supply voltage as the USB-Blaster, MasterBlaster (VIO pin), ByteBlaster II or ByteBlasterMV cable.

(2) Connect the nCONFIG and MSEL[1..0] pins to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect the nCONFIG pin to V_{CC}, and the MSEL[1..0] pins to ground. In addition, pull DCLK and DATA0 to either high or low, whichever is convenient on your board.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. Refer to the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value. In the ByteBlasterMV cable, this pin is a no connect. In the USB-Blaster and ByteBlaster II cable, this pin is connected to nCE when it is used for AS programming, otherwise it is a no connect.

(4) nCE must be connected to ground or driven low for successful JTAG configuration.

Connect the nCE pin to GND or pull it low during JTAG configuration. In multiple device AS and PS configuration chains, connect the first device's nCE pin to GND and connect its nCEO pin to the nCE pin of the next device in the chain or you can use it as a user I/O pin after configuration.

After the first device completes configuration in a multiple device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. Therefore, if these devices are also in a JTAG chain, you should make sure the nCE pins are connected to GND during JTAG configuration or that the devices are JTAG configured in the same order as the configuration chain. As long as the devices are JTAG configured in the same order as the multiple device configuration chain, the nCEO pin of the previous device drives the nCE pin of the next device low when it has successfully been JTAG configured.

The Quartus II software sets the Cyclone II device nCEO pin as an output pin driving to ground by default. If the nCEO pin inputs to the next device's nCE pin, make sure that the nCEO pin is not used as a user I/O pin after configuration.

Other Altera devices that have JTAG support can be placed in the same JTAG chain for device programming and configuration.

For more information on configuring multiple Altera devices in the same configuration chain, see the *Configuring Mixed Altera FPGA Chains* chapter in the *Configuration Handbook*.

Jam STAPL

Jam STAPL, JEDEC standard JESD-71, is a standard file format for in-system programmability (ISP). Jam STAPL supports programming or configuration of programmable devices and testing of electronic systems using the IEEE 1149.1 JTAG interface. Jam STAPL is a freely licensed open standard. The Jam player provides an interface for manipulating the IEEE Std. 1149.1 JTAG TAP state machine.

For more information on JTAG and Jam STAPL in embedded environments, see *AN 122: Using Jam STAPL for ISP & ICR via an Embedded Processor*. To download the Jam player, go to the Altera web site (**www.altera.com**).

Configuring Cyclone II FPGAs with JRunner

JRunner is a software driver that allows you to configure Cyclone II devices through the ByteBlaster II or ByteBlasterMV cables in JTAG mode. The programming input file supported is in **.rbf** format. JRunner also requires a Chain Description File (**.cdf**) generated by the Quartus II software. JRunner is targeted for embedded JTAG configuration. The source code has been developed for the Windows NT operating system (OS). You can customize the code to make it run on your embedded platform.

The RBF file used by the JRunner software driver can not be a compressed RBF file because JRunner uses JTAG-based configuration. During JTAG-based configuration, the real-time decompression feature is not available.

For more information on the JRunner software driver, see *JRunner Software Driver: An Embedded Solution for PLD JTAG Configuration* and the source files on the Altera web site.

Combining JTAG & Active Serial Configuration Schemes

You can combine the AS configuration scheme with JTAG-based configuration. Set the MSEL[1..0] pins to 00 (AS mode) or 10 (Fast AS mode)in this setup, which uses two 10-pin download cable headers on the board. The first header programs the serial configuration device in the system via the AS programming interface, and the second header configures the Cyclone II directly via the JTAG interface.

If you try configuring the device using both schemes simultaneously, JTAG configuration takes precedence and AS configuration is terminated.

When a blank serial configuration device is attached to Cyclone II device, turn on the **Halt on-chip configuration controller** option under the Tools menu by clicking **Options**. The Options dialog box appears. In the **Category** list, select **Programmer** before starting the JTAG configuration with the Quartus II programmer. This option stops the AS reconfiguration loop from a blank serial configuration device before starting the JTAG configuration. This includes using the Serial Flash Loader IP because JTAG is used for configuring the Cyclone II device. Users do not need to recompile their Quartus II designs after turning on this Option.

Programming Serial Configuration Devices In-System Using the JTAG Interface

Cyclone II devices in a single device chain or in a multiple device chain support in-system programming of a serial configuration device using the JTAG interface via the serial flash loader design. The board's intelligent host or download cable can use the four JTAG pins on the Cyclone II device to program the serial configuration device in system, even if the host or download cable cannot access the configuration device's configuration pins (DCLK, DATA, ASDI, and nCS pins).

The serial flash loader design is a JTAG-based in-system programming solution for Altera serial configuration devices. The serial flash loader is a bridge design for the FPGA that uses its JTAG interface to access the EPCS JIC (JTAG Indirect Configuration Device Programming) file and then uses the AS interface to program the EPCS device. Both the JTAG interface and AS interface are bridged together inside the serial flash loader design.

In a multiple device chain, you only need to configure the master Cyclone II device which is controlling the serial configuration device. The slave devices in the multiple device chain which are configured by the serial configuration device do not need to be configured when using this

feature. To use this feature successfully, set the MSEL[1..0] pins of the master Cyclone II device to select the AS configuration scheme or fast AS configuration scheme (see Table 13–1).

The Quartus II software version 4.1 and higher supports serial configuration device ISP through an FPGA JTAG interface using a JIC file.

The serial configuration device in-system programming through the Cyclone II JTAG interface has three stages, which are described in the following sections.

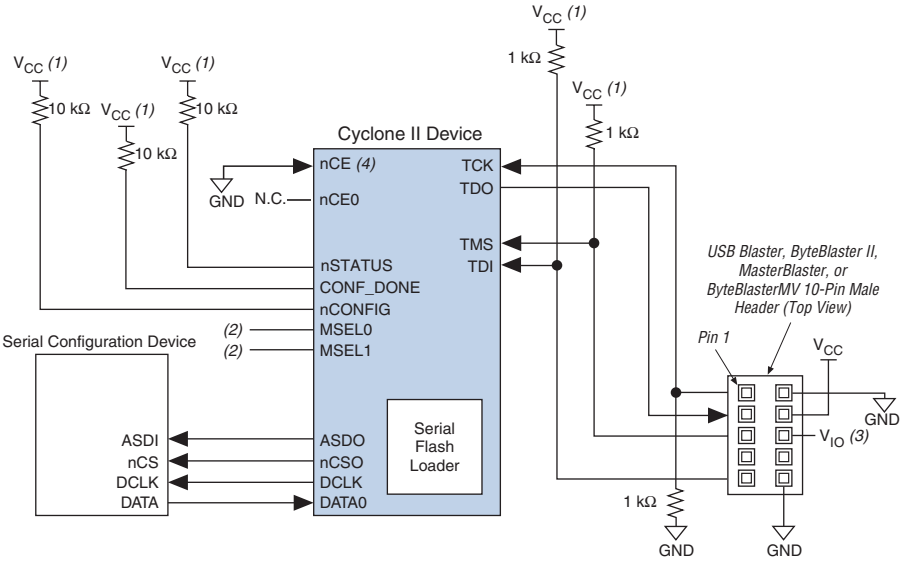
Loading the Serial Flash Loader Design

The serial flash loader design is a design inside the Cyclone II device that bridges the JTAG interface and AS interface inside the Cyclone II device using glue logic.

The intelligent host uses the JTAG interface to configure the master Cyclone II device with a serial flash loader design. The serial flash loader design allows the master Cyclone II device to control the access of four serial configuration device pins, also known as the Active Serial Memory Interface (ASMI) pins, through the JTAG interface. The ASMI pins are the serial clock input (DCLK), serial data output (DATA), AS data input (ASDI), and an active-low chip select (nCS) pins.

If you configure a master Cyclone II device with a serial flash loader design, the master Cyclone II device can enter user mode even though the slave devices in the multiple device chain are not being configured. The master Cyclone II device can enter user mode with a serial flash loader design even though the CONF_DONE signal is externally held low by the other slave devices in chain. Figure 13–25 shows the JTAG configuration of a single Cyclone II device with a serial flash loader design.

Figure 13–25. JTAG Configuration of a Single Device Using a Download Cable



Notes to Figure 13–25:

(1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster (V_{IO} pin), ByteBlaster II, or ByteBlasterMV cable.

(2) The `nCONFIG`, `MSEL[1..0]` pins should be connected to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect `nCONFIG` to V_{CC}, and `MSEL[1..0]` to ground. Pull `DCLK` either high or low, whichever is convenient on your board.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. Refer to the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value. In the ByteBlasterMV cable, this pin is a no connect. In the USB Blaster and ByteBlaster II cables, this pin is connected to `nCE` when it is used for active serial programming, otherwise it is a no connect.

(4) `nCE` must be connected to GND or driven low for successful JTAG configuration.

ISP of Serial Configuration Device

In the second stage, the serial flash loader design in the master Cyclone II device allows you to write the configuration data for the device chain into the serial configuration device by using the Cyclone II JTAG interface. The JTAG interface sends the programming data for the serial configuration device to the Cyclone II device first. The Cyclone II device then uses the ASMI pins to transmit the data to the serial configuration device.

Reconfiguration

After all the configuration data is written into the serial configuration device successfully, the Cyclone II device does not reconfigure by itself. The intelligent host issues the PULSE_NCONFIG JTAG instruction to initialize the reconfiguration process. During reconfiguration, the master Cyclone II device is reset and the serial flash loader design no longer exists in the Cyclone II device and the serial configuration device configures all the devices in the chain with your user design.

Device Configuration Pins

This section describes the connections and functionality of all the configuration related pins on the Cyclone II device. Table 13–11 describes the dedicated configuration pins, which are required to be connected properly on your board for successful configuration. Some of these pins may not be required for your configuration schemes.

Table 13–11. Dedicated Configuration Pins on the Cyclone II Device (Part 1 of 5)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
MSEL[1..0]	N/A	All	Input	This pin is a two-bit configuration input that sets the Cyclone II device configuration scheme. See Table 13–1 for the appropriate settings. You must connect these pins to V_{CCIO} or ground. The MSEL[1..0] pins have 9-kΩ internal pull-down resistors that are always active.
nCONFIG	N/A	All	Input	This pin is a configuration control input. If this pin is pulled low during user mode, the FPGA loses its configuration data, enters a reset state, and tri-states all I/O pins. Transitioning this pin high initiates a reconfiguration. If your configuration scheme uses an enhanced configuration device or EPC2 device, you can connect the nCONFIG pin directly to V_{CC} or to the configuration device's nINIT_CONF pin. The input buffer on this pin supports hysteresis using Schmitt trigger circuitry.

Table 13–11. Dedicated Configuration Pins on the Cyclone II Device (Part 2 of 5)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nSTATUS	N/A	All	Bidirectional open-drain	The Cyclone II device drives nSTATUS low immediately after power-up and releases it after the POR time. This pin provides a status output and input for the Cyclone II device. If the Cyclone II device detects an error during configuration, it drives the nSTATUS pin low to stop configuration. If an external source (for example, another Cyclone II device) drives the nSTATUS pin low during configuration or initialization, the target device enters an error state. Driving nSTATUS low after configuration and initialization does not affect the configured device. If your design uses a configuration device, driving nSTATUS low causes the configuration device to attempt to configure the FPGA, but since the FPGA ignores transitions on nSTATUS in user mode, the FPGA does not reconfigure. To initiate a reconfiguration, pull the nCONFIG pin low. The enhanced configuration devices' and EPC2 devices' OE and nCS pins are connected to the Cyclone II device's nSTATUS and CONF_DONE pins, respectively, and have optional internal programmable pull-up resistors. If you use these internal pull-up resistors on the enhanced configuration device, do not use external 10-kΩ pull-up resistors on these pins. When using EPC2 devices, you should only use external 10-kΩ pull-up resistors. The input buffer on this pin supports hysteresis using Schmitt trigger circuitry.

Table 13–11. Dedicated Configuration Pins on the Cyclone II Device (Part 3 of 5)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
CONF_DONE	N/A	All	Bidirectional open-drain	This pin is a status output and input. The target Cyclone II device drives the CONF_DONE pin low before and during configuration. Once the Cyclone II device receives all the configuration data without error and the initialization cycle starts, it releases CONF_DONE. Driving CONF_DONE low during user mode does not affect the configured device. Do not drive CONF_DONE low before the device enters user mode. After the Cyclone II device receives all the data, the CONF_DONE pin transitions high, and the device initializes and enters user mode. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize. Driving CONF_DONE low after configuration and initialization does not affect the configured device. The enhanced configuration devices' and EPC2 devices' OE and nCS pins are connected to the Cyclone II device's nSTATUS and CONF_DONE pins, respectively, and have optional internal programmable pull-up resistors. If internal pull-up resistors on the enhanced configuration device are used, external 10-kΩ pull-up resistors should not be used on these pins. When using EPC2 devices, you should only use external 10-kΩ pull-up resistors. The input buffer on this pin supports hysteresis using Schmitt trigger circuitry.
nCE	N/A	All	Input	This pin is an active-low chip enable. The nCE pin activates the device with a low signal to allow configuration. The nCE pin must be held low during configuration, initialization, and user mode. In single device configuration, it should be tied low. In multiple device configuration, nCE of the first device is tied low while its nCEO pin is connected to nCE of the next device in the chain. The nCE pin must also be held low for successful JTAG programming of the FPGA. The input buffer on this pin supports hysteresis using Schmitt trigger circuitry.

Table 13–11. Dedicated Configuration Pins on the Cyclone II Device (Part 4 of 5)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nCEO	N/A if option is on. I/O if option is off.	All	Output	This pin is an output that drives low when device configuration is complete. In single device configuration, you can leave this pin floating or use it as a user I/O pin after configuration. In multiple device configuration, this pin inputs the next device's nCE pin. The nCEO of the last device in the chain can be left floating or used as a user I/O pin after configuration. If you use the nCEO pin to feed next device's nCE pin, use an external 10-kΩ pull-up resistor to pull the nCEO pin high to the V_{CCIO} voltage of its I/O bank to help the internal weak pull-up resistor. Use the Quartus II software to make this pin a user I/O pin.
ASDO	N/A in AS mode I/O in PS and JTAG mode	AS	Output	This pin sends a control signal from the Cyclone II device to the serial configuration device in AS mode and is used to read out configuration data. In AS mode, ASDO has an internal pull-up that is always active.
nCSO	N/A in AS mode I/O in PS and JTAG mode	AS	Output	This pin sends an output control signal from the Cyclone II device to the serial configuration device in AS mode that enables the configuration device. In AS mode, nCSO has an internal pull-up resistor that is always active.

Table 13–11. Dedicated Configuration Pins on the Cyclone II Device (Part 5 of 5)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
DCLK	N/A	PS, AS	Input (PS) Output (AS)	In PS configuration, DCLK is the clock input used to clock data from an external source into the target device. Data is latched into the Cyclone II device on the rising edge of DCLK. In AS mode, DCLK is an output from the Cyclone II device that provides timing for the configuration interface. In AS mode, DCLK has an internal pull-up that is always active. After configuration, this pin is tri-stated. If you are using a configuration device, it drives DCLK low after configuration is complete. If your design uses a control host, drive DCLK either high or low. Toggling this pin after configuration does not affect the configured device. The input buffer on this pin supports hysteresis using Schmitt trigger circuitry.
DATA0	N/A	All	Input	This is the data input pin. In serial configuration modes, bit-wide configuration data is presented to the target device on the DATA0 pin. In AS mode, DATA0 has an internal pull-up resistor that is always active. After configuration, EPC1 and EPC1441 devices tri-state this pin, while enhanced configuration and EPC2 devices drive this pin high. The input buffer on this pin supports hysteresis using Schmitt trigger circuitry.

Table 13–12 describes the optional configuration pins. If these optional configuration pins are not enabled in the Quartus II software, they are available as general-purpose user I/O pins. Therefore during configuration, these pins function as user I/O pins and are tri-stated with weak pull-up resistors.

Table 13–12. Optional Configuration Pins

Pin Name	User Mode	Pin Type	Description
CLKUSR	N/A if option is on. I/O if option is off.	Input	This is an optional user-supplied clock input that synchronizes the initialization of one or more devices. This pin is enabled by turning on the **Enable user-supplied start-up clock (CLKUSR)** option in the Quartus II software
INIT_DONE	N/A if option is on. I/O if option is off.	Output open-drain	This is a status pin that can be used to indicate when the device has initialized and is in user mode. When nCONFIG is low and during the beginning of configuration, the INIT_DONE pin is tri-stated and pulled high due to an external 10-kΩ pull-up resistor. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high and the FPGA enters user mode. Thus, the monitoring circuitry must be able to detect a low-to-high transition. This pin is enabled by turning on the **Enable INIT_DONE output** option in the Quartus II software.
DEV_OE	N/A if option is on. I/O if option is off.	Input	Optional pin that allows the user to override all tri-states on the device. When this pin is driven low, all I/O pins are tri-stated. When this pin is driven high, all I/O pins behave as programmed. This pin is enabled by turning on the **Enable device-wide output enable (DEV_OE)** option in the Quartus II software.
DEV_CLRn	N/A if option is on. I/O if option is off.	Input	Optional pin that allows you to override all clears on all device registers. When this pin is driven low, all registers are cleared. When this pin is driven high, all registers behave as programmed. This pin is enabled by turning on the **Enable device-wide reset (DEV_CLRn)** option in the Quartus II software.

Table 13–13 describes the dedicated JTAG pins. JTAG pins must be kept stable before and during configuration to prevent accidental loading of JTAG instructions. The TCK pin has a weak internal pull-down resistor and the TDI and TMS JTAG input pins have weak internal pull-up resistors.

Table 13–13. Dedicated JTAG Pins

Pin Name	User Mode	Pin Type	Description
TDI	N/A	Input	Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}. The input buffer on this pin supports hysteresis using Schmitt trigger circuitry.
TDO	N/A	Output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by leaving this pin unconnected.
TMS	N/A	Input	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}. The input buffer on this pin supports hysteresis using Schmitt trigger circuitry.
TCK	N/A	Input	The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND. The input buffer on this pin supports hysteresis using Schmitt trigger circuitry.

Conclusion

Cyclone II devices can be configured in AS, PS or JTAG configuration schemes to fit your system's need. The AS configuration scheme supported by Cyclone II devices can now operate at a higher DCLK

frequency (up to 40 MHz), which reduces your configuration time. In addition, Cyclone II devices can receive a compressed configuration bitstream and decompress this data on-the-fly in the AS or PS configuration scheme, which further reduces storage requirements and configuration time.

Document Revision History

Table 13–14 shows the revision history for this document.

Table 13–14. Document Revision History

Date & Document Version	Changes Made	Summary of Changes
February 2007 v3.1	• Added document revision history. • Added *Note (1)* to Table 13–1. • Added *Note (1)* to Table 13–4. • Updated Figure 13–3. • Updated Figures 13–6 and 13–7. • Updated *Note (2)* to Figure 13–13. • Updated "Single Device PS Configuration Using a Configuration Device" section. • Updated *Note (2)* to Figure 13–14. • Updated *Note (2)* to Figure 13–15. • Updated *Note (2)* to Figure 13–16. • Updated *Note (2)* to Figure 13–17. • Updated *Note (4)* to Figure 13–21. • Updated *Note (2)* to Figure 13–25.	• Changed unit 'kw' to 'kΩ' in Figures 13–6 and 13–7. • Added note about serial configuration devices supporting 20 MHz and 40 MHz DCLK. • Added infomation about the need for a resistor on nCONFIG if reconfiguration is required. • Added information about MSEL[1..0] internal pull-down resistor value.
July 2005 v2.0	• Updated "Configuration Stage" section. • Updated "PS Configuration Using a Download Cable" section. • Updated Figures 13–8, 13–12, and 13–18.	
November 2004 v1.1	• Updated "Configuration Stage" section in "Single Device AS Configuration" section. • Updated "Initialization Stage" section in "Single Device AS Configuration" section. • Updated Figure 13–8. • Updated "Initialization Stage" section in "Single Device PS Configuration Using a MAX II Device as an External Host" section. • Updated Table 13–7. • Updated "Single Device PS Configuration Using a Configuration Device" section. • Updated "Initialization Stage" section in "Single Device PS Configuration Using a Configuration Device" section. • Updated Figure 13–18. • Updated "Single Device JTAG Configuration" section.	
June 2004 v1.0	Added document to the Cyclone II Device Handbook.	

13. Configuring Cyclone FPGAs

C51013-1.8

Introduction

You can configure Cyclone® FPGAs using one of several configuration schemes, including the active serial (AS) configuration scheme. This scheme is used with the low cost serial configuration devices. Passive serial (PS) and Joint Test Action Group (JTAG)-based configuration schemes are also supported by Cyclone FPGAs. Additionally, Cyclone FPGAs can receive a compressed configuration bit stream and decompress this data in real-time, reducing storage requirements and configuration time.

This chapter describes how to configure Cyclone devices using each of the three supported configuration schemes.

For more information about setting device configuration options or generating configuration files, refer to the *Software Settings* section in volume 2 of the *Configuration Handbook*.

Device Configuration Overview

Cyclone FPGAs use SRAM cells to store configuration data. Since SRAM memory is volatile, configuration data must be downloaded to Cyclone FPGAs each time the device powers up. You can download configuration data to Cyclone FPGAs using the AS, PS, or JTAG interfaces (see Table 13–1).

Table 13–1. Cyclone FPGA Configuration Schemes

Configuration Scheme	Description
Active serial (AS) configuration	Configuration using: • Serial configuration devices (EPCS1, EPCS4, and EPCS16)
Passive serial (PS) configuration	Configuration using: • Enhanced configuration devices (EPC4, EPC8, and EPC16) • EPC2, EPC1 configuration devices • Intelligent host (microprocessor) • Download cable
JTAG-based configuration	Configuration via JTAG pins using: • Download cable • Intelligent host (microprocessor) • Jam™ Standard Test and Programming Language (STAPL) • Ability to use SignalTap® II Embedded Logic Analyzer.

You can select a Cyclone FPGA configuration scheme by driving its MSEL1 and MSEL0 pins either high (1) or low (0), as shown in Table 13–2. If your application only requires a single configuration mode, the MSEL pins can be connected to V_{CC} (the I/O bank's V_{CCIO} voltage where the MSEL pin resides) or to ground. If your application requires more than one configuration mode, the MSEL pins can be switched after the FPGA has been configured successfully. Toggling these pins during user mode does not affect the device operation. However, the MSEL pins must be valid before initiating reconfiguration.

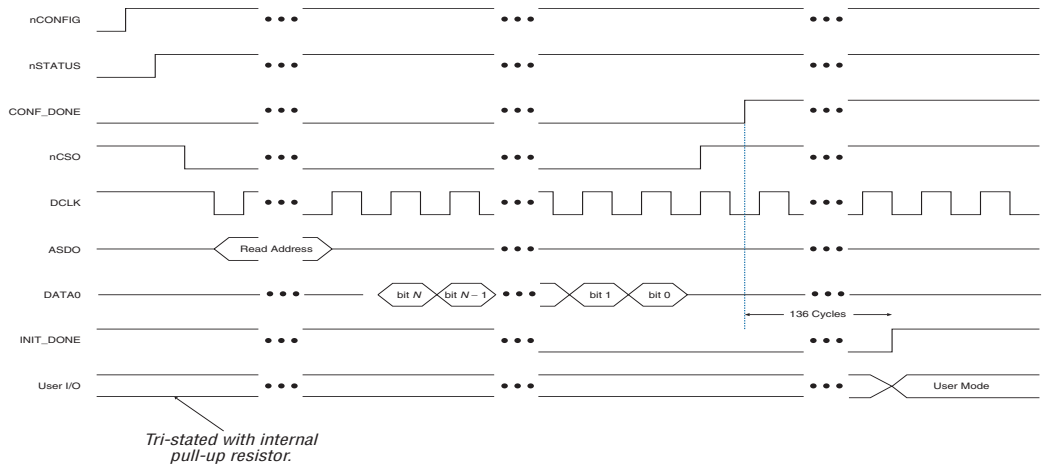
Table 13–2. Selecting Cyclone Configuration Schemes

MSEL1	MSEL0	Configuration Scheme
0	0	AS
0	1	PS
0	1	JTAG-based *(1)*

Note to Table 13–2:
(1) JTAG-based configuration takes precedence over other schemes, which means that MSEL pin settings are ignored.

After configuration, Cyclone FPGAs will initialize registers and I/O pins, then enter user mode and function as per the user design. Figure 13–1 shows an AS configuration waveform.

Figure 13–1. AS Configuration Waveform



You can configure Cyclone FPGAs using the 3.3-, 2.5-, 1.8-, or 1.5-V LVTTL I/O standard on configuration and JTAG input pins. These devices do not feature a VCCSEL pin; therefore, you should connect the VCCIO pins of the I/O banks containing configuration or JTAG pins according to the I/O standard specifications.

Table 13–3 summarizes the approximate uncompressed configuration file size for each Cyclone FPGA. To calculate the amount of storage space required for multi-device configurations, add the file size of each device together.

Table 13–3. Cyclone Raw Binary File (.rbf) Sizes

Device	Data Size (Bits)	Data Size (Bytes)
EP1C3	627,376	78,422
EP1C4	924,512	115,564
EP1C6	1,167,216	145,902
EP1C12	2,323,240	290,405
EP1C20	3,559,608	435,000

You should only use the numbers in Table 13–3 to estimate the configuration file size before design compilation. Different file formats, such as **.hex** or **.ttf** files, have different file sizes. For any specific version of the Quartus® II software, any design targeted for the same device has the same uncompressed configuration file size. If compression is used, the file size can vary after each compilation.

Data Compression

Cyclone FPGAs are the first FPGAs to support decompression of configuration data. This feature allows you to store compressed configuration data in configuration devices or other memory, and transmit this compressed bit stream to Cyclone FPGAs. During configuration, the Cyclone FPGA decompresses the bit stream in real time and programs its SRAM cells.

Cyclone FPGAs support compression in the AS and PS configuration schemes. Compression is not supported for JTAG-based configuration.

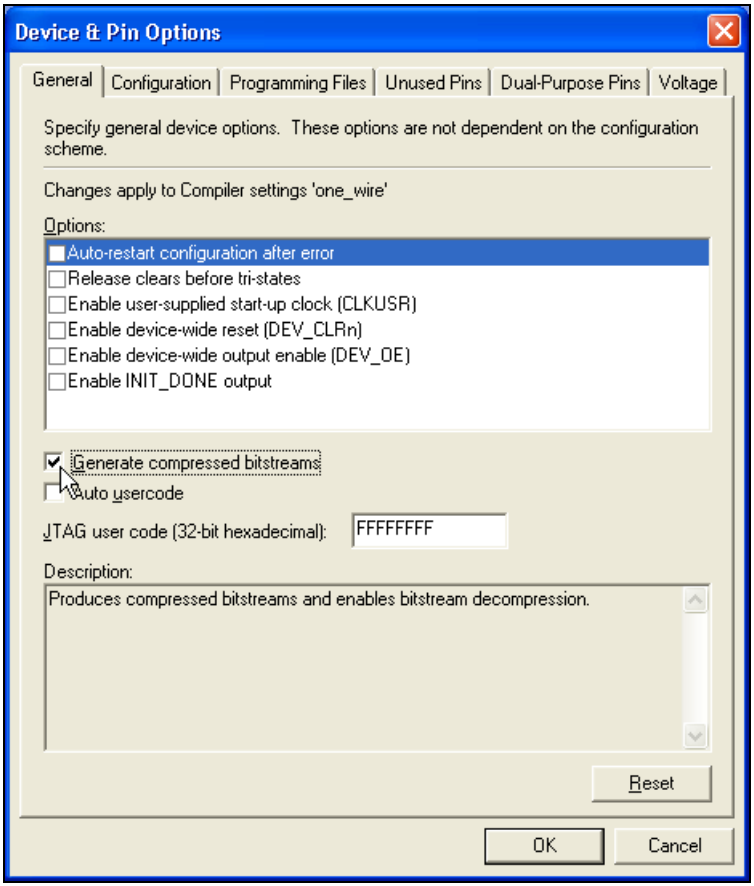
Preliminary data indicates that compression reduces configuration bit stream size by 35 to 60%.

When you enable compression, the Quartus II software generates configuration files with compressed configuration data. This compression reduces the storage requirements in the configuration device or flash, and decreases the time needed to transmit the bit stream to the Cyclone FPGA.

There are two methods to enable compression for Cyclone bitstreams: before design compilation (in the Compiler Settings menu) and after design compilation (in the **Convert Programming Files** window).

To enable compression in the project's compiler settings, select **Device** under the Assignments menu to bring up the settings window. After selecting your Cyclone device open the **Device and Pin Options** window, and in the **General** settings tab enable the check box for **Generate compressed bitstreams** (as shown in Figure 13–2).

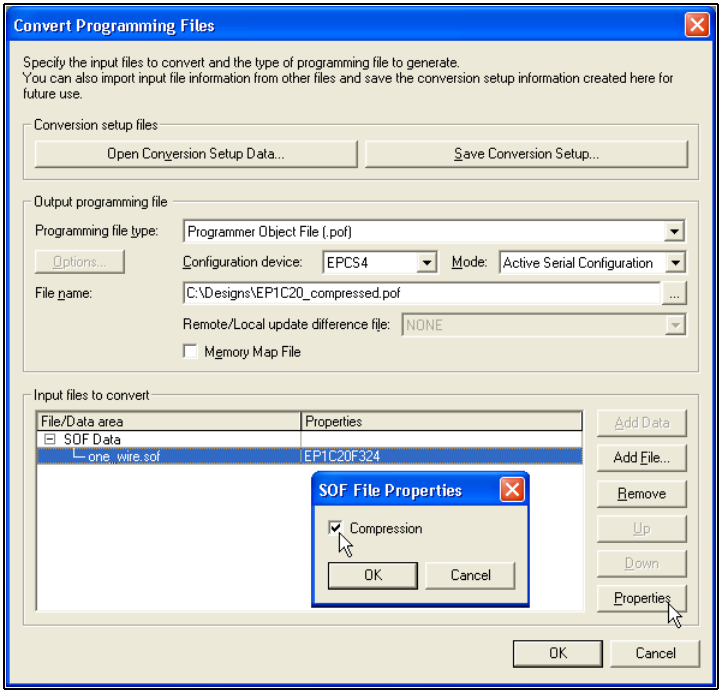
Figure 13–2. Enabling Compression for Cyclone Bitstreams in Compiler Settings



Compression can also be enabled when creating programming files from the **Convert Programming Files** window. See Figure 13–3.

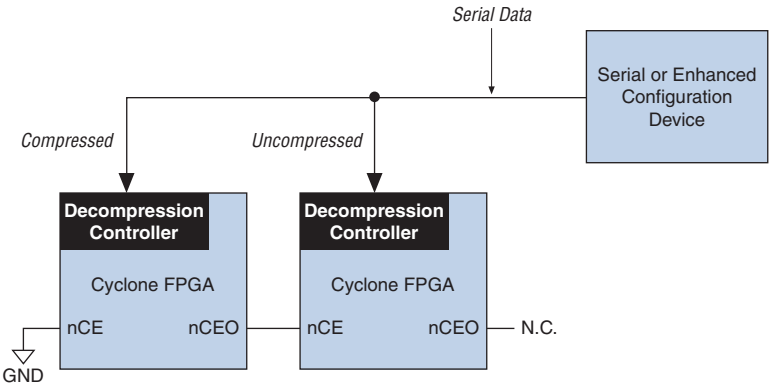
1. Click **Convert Programming Files** (File menu).
2. Select the programming file type (POF, SRAM HEXOUT, RBF, or TTF).
3. For POF output files, select a configuration device.
4. Select **Add File** and add a Cyclone SOF file(s).
5. Select the name of the file you added to the **SOF Data** area and click **Properties**.
6. Turn on **Compression**.

Figure 13–3. Enabling Compression for Cyclone Bitstreams in Convert Programming Files



When multiple Cyclone devices are cascaded, the compression feature can be selectively enabled for each device in the chain. Figure 13–4 depicts a chain of two Cyclone FPGAs. The first Cyclone FPGA has the compression feature enabled and therefore receives a compressed bit stream from the configuration device. The second Cyclone FPGA has the compression feature disabled and receives uncompressed data.

Figure 13–4. Compressed and Uncompressed Configuration Data in the Same Programming File *Note (1)*



Note to Figure 13–4:

(1) The first device in the chain should be set up in AS configuration mode (`MSEL[1..0]="00"`). The remaining devices in the chain must be set up in PS configuration mode (`MSEL[1..0]="01"`).

You can generate programming files for this setup from the **Convert Programming Files** window (File menu) in the Quartus II software.

The decompression feature supported by Cyclone FPGAs is separate from the decompression feature in enhanced configuration devices (EPC16, EPC8, and EPC4 devices). The data compression feature in the enhanced configuration devices allows them to store compressed data and decompress the bit stream before transmitting to the target devices. When using Cyclone FPGAs with enhanced configuration devices, Altera recommends using compression on one of the devices, not both (preferably the Cyclone FPGA since transmitting compressed data reduces configuration time).

Configuration Schemes

This section describes the various configuration schemes you can use to configure Cyclone FPGAs. Descriptions include an overview of the protocol, pin connections, and timing information. The schemes discussed are:

- AS configuration (serial configuration devices)
- PS configuration
- JTAG-based configuration

Active Serial Configuration (Serial Configuration Devices)

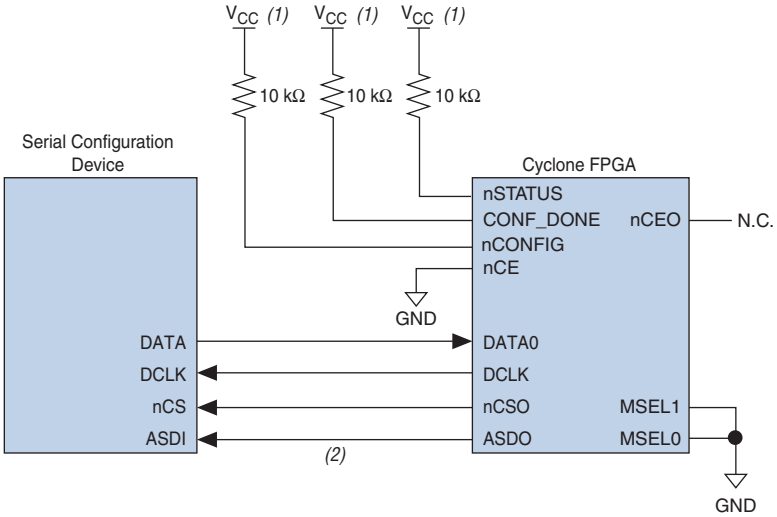
In the AS configuration scheme, Cyclone FPGAs are configured using the new serial configuration devices. These configuration devices are low cost devices with non-volatile memory that feature a simple four-pin interface and a small form factor. These features make serial configuration devices an ideal solution for configuring the low-cost Cyclone FPGAs.

For more information on programming serial configuration devices, refer to the Cyclone Literature web page at www.altera.com and the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet*.

Serial configuration devices provide a serial interface to access configuration data. During device configuration, Cyclone FPGAs read configuration data via the serial interface, decompress data if necessary, and configure their SRAM cells. This scheme is referred to as an AS configuration scheme because the FPGA controls the configuration interface. This scheme is in contrast to the PS configuration scheme where the configuration device controls the interface.

Serial configuration devices have a four-pin interface: serial clock input (`DCLK`), serial data output (`DATA`), AS data input (`ASDI`), and an active-low chip select (`nCS`). This four-pin interface connects to Cyclone FPGA pins as shown in Figure 13–5.

Figure 13–5. AS Configuration of a Single Cyclone FPGA



Notes to Figure 13–5:
(1) Connect the pull-up resistors to a 3.3-V supply.
(2) Cyclone FPGAs use the ASDO to ASDI path to control the configuration device.

Connecting the MSEL[1..0] pins to 00 selects the AS configuration scheme. The Cyclone chip enable signal, nCE, must also be connected to ground or driven low for successful configuration.

During system power up, both the Cyclone FPGA and serial configuration device enter a power-on reset (POR) period. As soon as the Cyclone FPGA enters POR, it drives nSTATUS low to indicate it is busy and drives CONF_DONE low to indicate that it has not been configured. After POR, which typically lasts 100 ms, the Cyclone FPGA releases nSTATUS and enters configuration mode when this signal is pulled high by the external 10-kΩ resistor. Once the FPGA successfully exits POR, all user I/O pins are tri-stated. Cyclone devices have weak pull-up resistors on the user I/O pins which are on before and during configuration.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *DC and Switching Characteristics* chapter in the *Cyclone Device Handbook*.

The serial clock (DCLK) generated by the Cyclone FPGA controls the entire configuration cycle (see Figure 13–1 on page 13–2) and this clock signal provides the timing for the serial interface. Cyclone FPGAs use an

internal oscillator to generate DCLK. After configuration, this internal oscillator is turned off. Table 13–4 shows the active serial DCLK output frequencies.

Table 13–4. Active Serial DCLK Output Frequency

Minimum	Typical	Maximum	Units
14	17	20	MHz

The serial configuration device latches input/control signals on the rising edge of DCLK and drives out configuration data on the falling edge. Cyclone FPGAs drive out control signals on the falling edge of DCLK and latch configuration data on the falling edge of DCLK.

In configuration mode, the Cyclone FPGA enables the serial configuration device by driving its nCSO output pin low that is connected to the chip select (nCS) pin of the configuration device. The Cyclone FPGA's serial clock (DCLK) and serial data output (ASDO) pins send operation commands and read-address signals to the serial configuration device. The configuration device provides data on its serial data output (DATA) pin that is connected to the DATA0 input on Cyclone FPGAs.

After the Cyclone FPGA receives all configuration bits, it releases the open-drain CONF_DONE pin allowing the external 10-kΩ resistor to pull this signal to a high level. Initialization begins only after the CONF_DONE line reaches a high level. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

You can select the clock used for initialization by using the **User Supplied Start-Up Clock** option in the Quartus II software. The Quartus II software uses the 10-MHz (typical) internal oscillator (separate from the AS internal oscillator) by default to initialize the Cyclone FPGA. After initialization, the internal oscillator is turned off. When you enable the **User Supplied Start-Up Clock** option, the software uses the CLKUSR pin as the initialization clock. Supplying a clock on the CLKUSR pin does not affect the configuration process. After all configuration data is accepted and the CONF_DONE signal goes high, Cyclone devices require 136 clock cycles to initialize properly.

An optional INIT_DONE pin is available. This pin signals the end of initialization and the start of user mode with a low-to-high transition. The **Enable INIT_DONE output** option is available in the Quartus II software. If the INIT_DONE pin is used, it is high due to an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the

INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. This low-to-high transition signals that the FPGA has entered user mode. In user mode, the user I/O pins do not have weak pull-ups and functions as assigned in your design.

If an error occurs during configuration, the Cyclone FPGA asserts the nSTATUS signal low indicating a data frame error, and the CONF_DONE signal stays low. With the **Auto-Restart Configuration on Frame Error** option enabled in the Quartus II software, the Cyclone FPGA resets the configuration device by pulsing nCSO, releases nSTATUS after a reset time-out period (about 30 µs), and retries configuration. If this option is turned off, the system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 40 µs to restart configuration. After successful configuration, the CONF_DONE signal is tri-stated by the target device and then pulled high by the pull-up resistor.

All AS configuration pins, DATA0, DCLK, nCSO, and ASDO, have weak internal pull-up resistors. These pull-up resistors are always active.

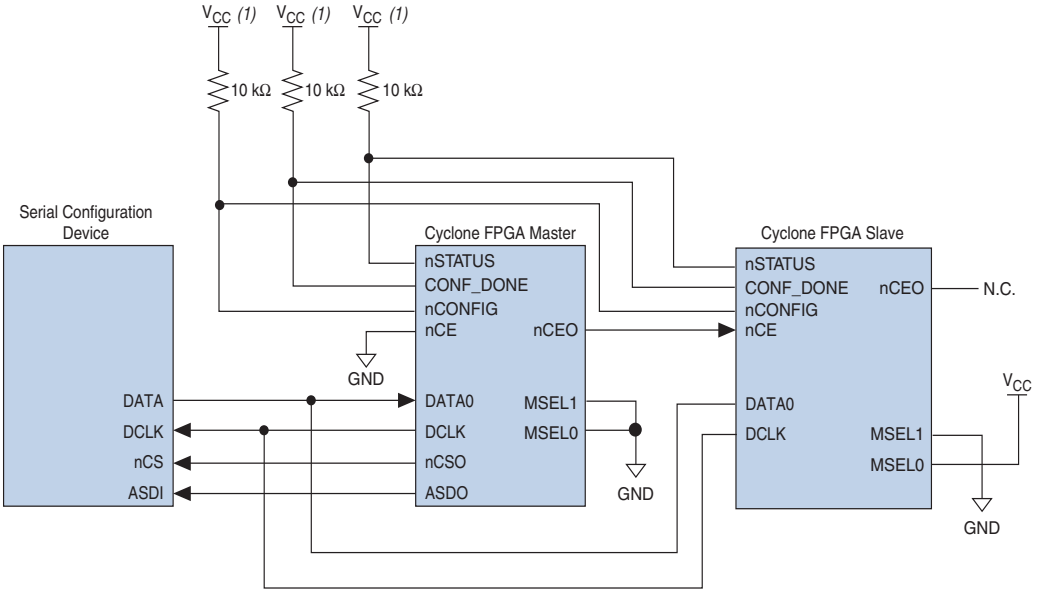
When the Cyclone FPGA is in user mode, you can initiate reconfiguration by pulling the nCONFIG pin low. The nCONFIG pin should be low for at least 40 µs. When nCONFIG is pulled low, the FPGA also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Once nCONFIG returns to a logic high level and nSTATUS is released by the Cyclone FPGA, reconfiguration begins.

Configuring Multiple Devices (Cascading)

You can configure multiple Cyclone FPGAs using a single serial configuration device. You can cascade multiple Cyclone FPGAs using the chip-enable (nCE) and chip-enable-out (nCEO) pins. The first device in the chain must have its nCE pin connected to ground. You must connect its nCEO pin to the nCE pin of the next device in the chain. When the first device captures all of its configuration data from the bit stream, it drives the nCEO pin low enabling the next device in the chain. You must leave the nCEO pin of the last device unconnected. The nCONFIG, nSTATUS, CONF_DONE, DCLK, and DATA0 pins of each device in the chain are connected (see Figure 13–6).

This first Cyclone FPGA in the chain is the configuration master and controls configuration of the entire chain. You must connect its MSEL pins to select the AS configuration scheme. The remaining Cyclone FPGAs are configuration slaves and you must connect their MSEL pins to select the PS configuration scheme. Figure 13–6 shows the pin connections for this setup.

Figure 13–6. Configuring Multiple Devices Using a Serial Configuration Device (AS)



Note to Figure 13–6:
(1) Connect the pull-up resistors to a 3.3-V supply.

As shown in Figure 13–6, the nSTATUS and CONF_DONE pins on all target FPGAs are connected together with external pull-up resistors. These pins are open-drain bidirectional pins on the FPGAs. When the first device asserts nCEO (after receiving all of its configuration data), it releases its CONF_DONE pin. But the subsequent devices in the chain keep this shared CONF_DONE line low until they have received their configuration data. When all target FPGAs in the chain have received their configuration data and have released CONF_DONE, the pull-up resistor drives a high level on this line and all devices simultaneously enter initialization mode. If an error occurs at any point during configuration, the nSTATUS line is driven low by the failing FPGA. If you enable the **Auto Restart Configuration on Frame Error** option, reconfiguration of the entire chain begins after a reset time-out period (a maximum of 40 µs). If the option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low to restart configuration. The external system can pulse nCONFIG if it is under system control rather than tied to V_{CC}.

While you can cascade Cyclone FPGAs, serial configuration devices cannot be cascaded or chained together.

If the configuration bit stream size exceeds the capacity of a serial configuration device, you must select a larger configuration device and/or enable the compression feature. While configuring multiple devices, the size of the bit stream is the sum of the individual devices' configuration bit streams.

Configuring Multiple Devices with the Same Data

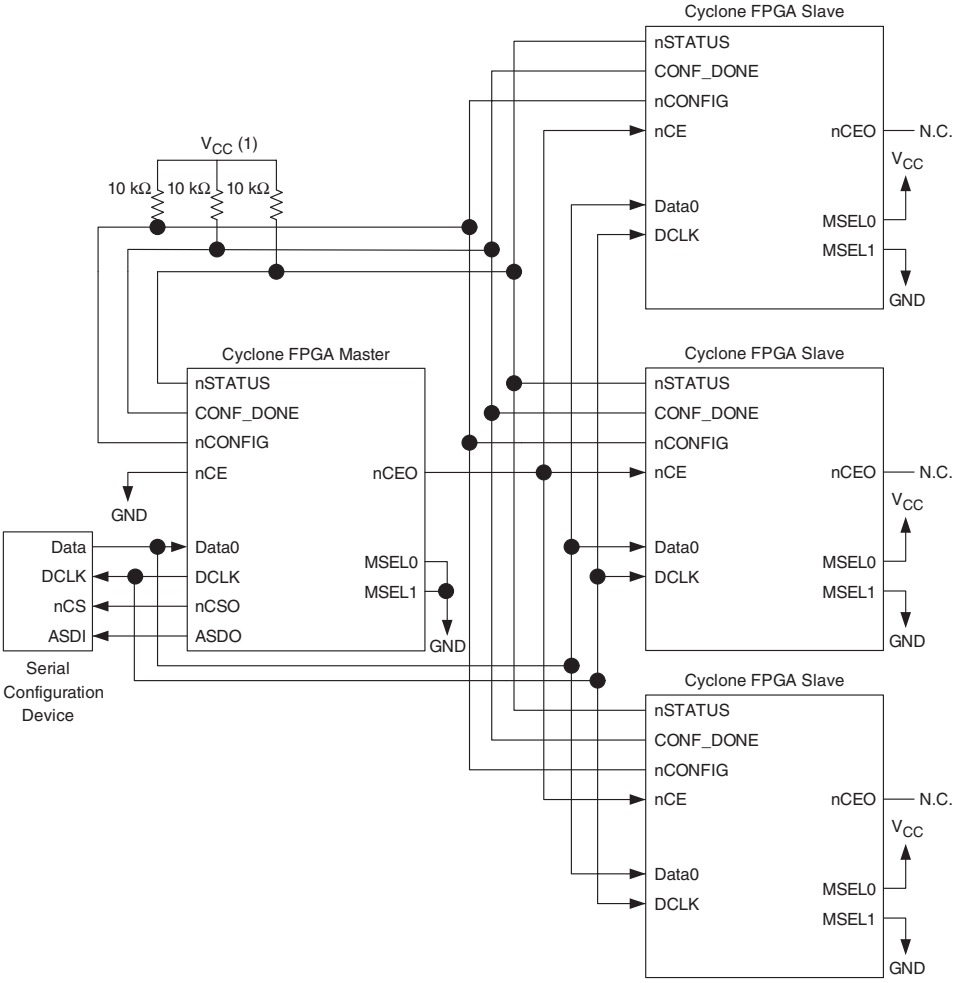
Certain applications require the configuration of multiple Cyclone devices with the same design through a configuration bit stream or SOF file. This can actually be done by two methods and they are shown below. For both methods, the serial configuration devices cannot be cascaded or chained together.

Method 1

For method 1, the serial configuration device stores two copies of the SOF file. The first copy configures the master Cyclone device, and the second copy configures all the remaining slave devices concurrently. The setup is similar to Figure 13–7 where the master is setup in AS mode (MSEL=00) and the slave devices are setup in PS mode (MSEL01).

To configure four identical Cyclone devices with the same SOF file, you could setup the chain similar to the example shown in Figure 13–6, except connect the three slave devices for concurrent configuration. The nCEO pin from the master device drives the nCE input pins on all three slave devices, and the DATA and DCLK pins connect in parallel to all four devices. During the first configuration cycle, the master device reads its configuration data from the serial configuration device while holding nCEO high. After completing its configuration cycle, the master drives nCE low and transmits the second copy of the configuration data to all three slave devices, configuring them simultaneously. The advantage of using the setup in Figure 13–7 is you can have a different SOF file for the Cyclone master device. However, all the Cyclone slave devices must be configured with the same SOF file.

Figure 13–7. Configuring Multiple Devices with the Same Design Using a Serial Configuration Device



Note to Figure 13–7:

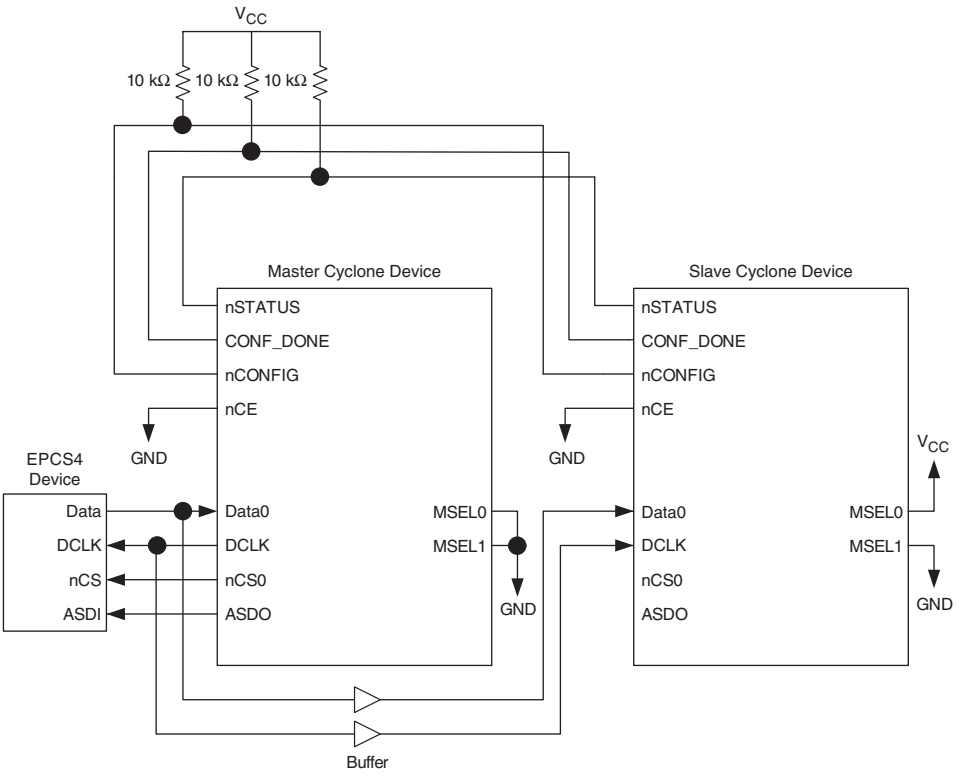
(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

Method 2

Method 2 configures multiple Cyclone devices with the same SOFs by storing only one copy of the SOF in the serial configuration device. This saves memory space in the serial configuration device for general-purpose use and may reduce costs. This method is shown in Figure 13–8 where the master device is set up in AS mode (MSLE=00), and

the slave devices are set up in PS mode (MSEL=01). You could set up one or more slave devices in the chain and all the slave devices are set up in the same way as the design shown in Figure 13–8.

Figure 13–8. Configuring Multiple Devices with the Same Design Using a Serial Configuration Device



In this setup, all the Cyclone devices in the chain are connected for concurrent configuration. This reduces the active serial configuration time because all the Cyclone devices are configured in only one configuration cycle. To achieve this, the nCE input pins on all the Cyclone devices are connected to ground and the nCEO output pins on all the Cyclone devices are left unconnected. The DATA and DCLK pins connect in parallel to all the Cyclone devices.

It is recommended to add a buffer before the DATA and DCLK output from the master Cyclone to avoid signal strength and signal integrity issues. The buffer should not significantly change the DATA-to-DCLK relationships or delay them with respect to other ASMI signals, which are

ASDI and nCS signals. Also, the buffer should only drive the slave Cyclone devices, so that the timing between the master Cyclone device and serial configuration device is unaffected.

This setup can support both compressed and uncompressed SOFs. Therefore, if the configuration bit stream size exceeds the capacity of a serial configuration device, you can enable the compression feature on the SOF used or you can select a larger serial configuration device.

Estimating Active Serial Configuration Time

Active serial configuration time is dominated by the time it takes to transfer data from the serial configuration device to the Cyclone FPGA. This serial interface is clocked by the Cyclone DCLK output (generated from an internal oscillator). As listed in Table 13–4, the DCLK minimum frequency is 14 MHz (71 ns). Therefore, the maximum configuration time estimate for an EP1C3 device (0.628 MBits of uncompressed data) is:

(0.628 MBits × 71 ns) = 47 ms.

The typical configuration time is 33 ms.

Enabling compression reduces the amount of configuration data that is transmitted to the Cyclone device, reducing configuration time. On average, compression reduces configuration time by 50%.

Programming Serial Configuration Devices

Serial configuration devices are non-volatile, flash-memory-based devices. You can program these devices in-system using the ByteBlaster™ II download cable. Alternatively, you can program them using the Altera Programming Unit (APU) or supported third-party programmers.

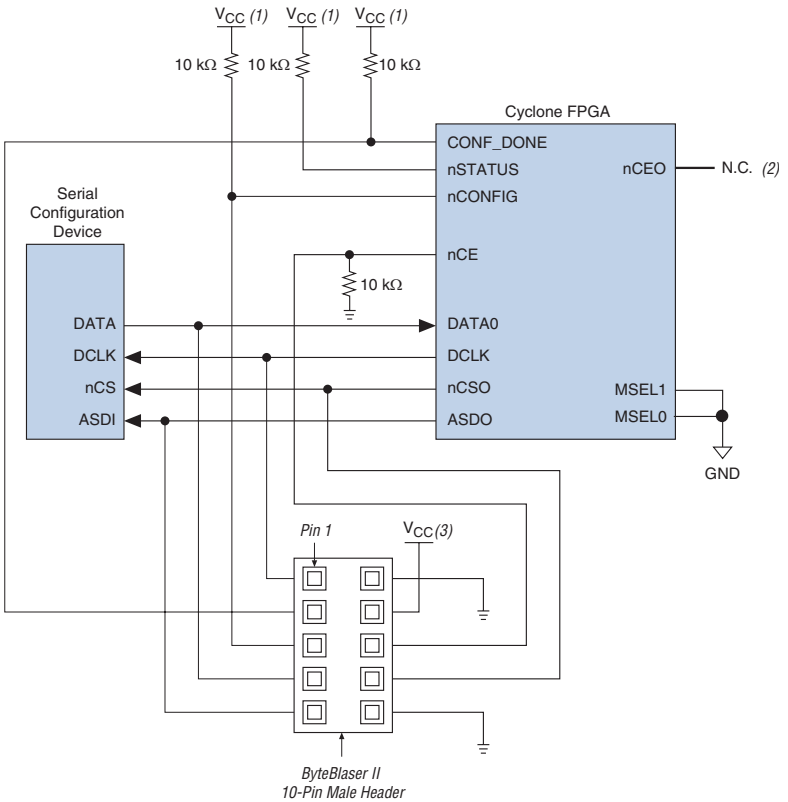
You can perform in-system programming of serial configuration devices via the AS programming interface. During in-system programming, the download cable disables FPGA access to the AS interface by driving the nCE pin high. Cyclone FPGAs are also held in reset by a low level on nCONFIG. After programming is complete, the download cable releases nCE and nCONFIG, allowing the pull-down and pull-up resistor to drive GND and VCC, respectively. Figure 13–9 shows the download cable connections to the serial configuration device.

For more information about the ByteBlaster II cable, refer to the *ByteBlaster II Download Cable User Guide*.

The serial configuration devices can be programmed in-system by an external microprocessor using SRunner. SRunner is a software driver developed for embedded serial configuration device programming that can be customized to fit in different embedded systems. The SRunner can read a Raw Programming Data file (.**rpd**) and write to the serial configuration devices. The programming time is comparable to the Quartus II software programming time.

For more information about SRunner, refer tothe *AN 418: SRunner: An Embedded Solution for Serial Configuration Device Programming* and the source code on the Altera website (www.altera.com).

Figure 13–9. In-System Programming of Serial Configuration Devices



Notes to Figure 13–9:

(1) Connect these pull-up resistors to 3.3-V supply.

(2) The nCEO pin is left unconnected.

(3) Power up the ByteBlaster II cable's V_{CC} with a 3.3-V supply.

You can program serial configuration devices by using the Quartus II software with the APU and the appropriate configuration device programming adapter. All serial configuration devices are offered in an eight-pin small outline integrated circuit (SOIC) package and can be programmed using the PLMSEPC-8 adapter.

In production environments, serial configuration devices can be programmed using multiple methods. Altera programming hardware (APU) or other third-party programming hardware can be used to program blank serial configuration devices before they are mounted onto PCBs. Alternatively, you can use an on-board microprocessor to program the serial configuration device in-system using C-based software drivers provided by Altera.

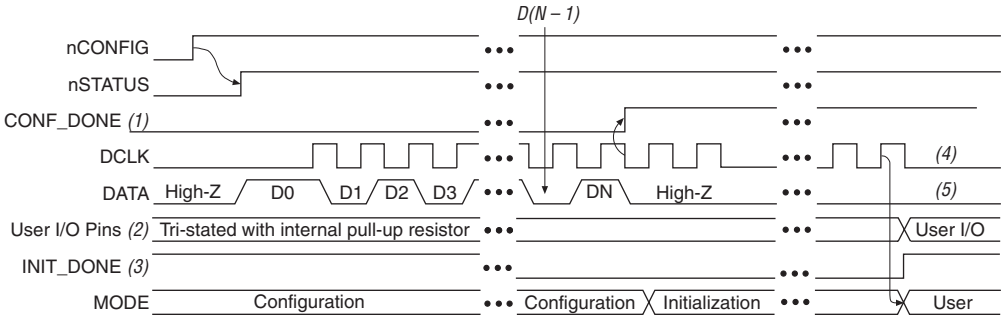
For more information on programming serial configuration devices, refer to the Cyclone Literature web page at www.altera.com and the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet*.

Device configuration options and how to create configuration files are discussed further in the *Software Settings* section in volume 2 of the *Configuration Handbook*.

Passive Serial Configuration

Cyclone FPGAs also feature the PS configuration scheme supported by all Altera FPGAs. In the PS scheme, an external host (configuration device, embedded processor, or host PC) controls configuration. Configuration data is clocked into the target Cyclone FPGAs via the DATA0 pin at each rising edge of DCLK. The configuration waveforms for this scheme are shown in Figure 13–10.

Figure 13–10. PS Configuration Cycle Waveform



Notes to Figure 13–10:

(1) During initial power up and configuration, CONF_DONE is low. After configuration, CONF_DONE goes high to indicate successful configuration. If the device is reconfigured, CONF_DONE goes low after nCONFIG is driven low.

(2) User I/O pins are tri-stated during configuration. Cyclone FPGAs also have a weak pull-up resistor on I/O pins during configuration. After initialization, the user I/O pins perform the function assigned in the user's design.

(3) When used, the optional INIT_DONE signal is high when nCONFIG is low before configuration and during the first 136 clock cycles of configuration.

(4) In user mode, DCLK should be driven high or low when using the PS configuration scheme. When using the AS configuration scheme, DCLK is a Cyclone output pin and should not be driven externally.

(5) In user mode, DATA0 should be driven high or low.

PS Configuration Using Configuration Device

In the PS configuration device scheme, nCONFIG is usually tied to V_{CC} (when using EPC16, EPC8, EPC4, or EPC2 devices, you can connect nCONFIG to nINIT_CONF). Upon device power-up, the target Cyclone FPGA senses the low-to-high transition on nCONFIG and initiates configuration. The target device then drives the open-drain CONF_DONE pin low, which in-turn drives the configuration device's nCS pin low. When exiting POR, both the target and configuration device release the open-drain nSTATUS pin (typically Cyclone POR lasts 100 ms).

Before configuration begins, the configuration device goes through a POR delay of up to 100 ms (maximum) to allow the power supply to stabilize. You must power the Cyclone FPGA before or during the POR time of the enhanced configuration device. During POR, the configuration device drives its OE pin low. This low signal delays configuration because the OE pin is connected to the target device's nSTATUS pin. When the target and configuration devices complete POR, they both release the nSTATUS to OE line, which is then pulled high by a pull-up resistor.

When configuring multiple devices, configuration does not begin until all devices release their OE or nSTATUS pins. When all devices are ready, the configuration device clocks out DATA and DCLK to the target devices using an internal oscillator.

After successful configuration, the Cyclone FPGA starts initialization using the 10-MHz internal oscillator as the reference clock. After initialization, this internal oscillator is turned off. The CONF_DONE pin is released by the target device and then pulled high by a pull-up resistor. When initialization is complete, the target Cyclone FPGA enters user mode. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

If an error occurs during configuration, the target device drives its nSTATUS pin low, resetting itself internally and resetting the configuration device. If you turn on the **Auto-Restart Configuration on Frame Error** option, the device reconfigures automatically if an error occurs. To set this option, select **Compiler Settings** (Processing menu), and click on the **Chips & Devices** tab. Select **Device and Pin Options**, and click on the **Configuration** tab.

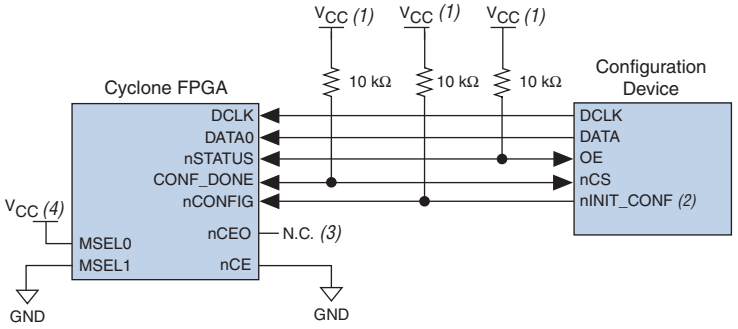
If the **Auto-Restart Configuration on Frame Error** option is turned off, the external system (configuration device or microprocessor) must monitor nSTATUS for errors and then pulse nCONFIG low to restart configuration. The external system can pulse nCONFIG if it is under system control rather than tied to V_{CC}. When configuration is complete, the target device releases CONF_DONE, which disables the configuration device by driving nCS high. The configuration device drives DCLK low before and after configuration.

In addition, if the configuration device sends all of its data and then detects that CONF_DONE has not gone high, it recognizes that the target device has not configured successfully. (For CONF_DONE to reach a high state, enhanced configuration devices wait for 64 DCLK cycles after the last configuration bit. EPC2 devices wait for 16 DCLK cycles.) In this case, the configuration device pulses its OE pin low for a few microseconds, driving the target device's nSTATUS pin low. If the **Auto-Restart Configuration on Frame Error** option is set in the Quartus II software, the target device resets and then releases its nSTATUS pin after a reset time-out period. When nSTATUS returns high, the configuration device reconfigures the target device.

You should not pull CONF_DONE low to delay initialization. Instead, use the Quartus II software's **User-Supplied Start-Up Clock** option to synchronize the initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain initialize together since their CONF_DONE pins are tied together.

CONF_DONE goes high during the first few clock cycles of initialization. Hence, when using the CLKUSR feature you would not see the CONF_DONE signal high until you start clocking CLKUSR. However, the device does retain configuration data and waits for these initialization clocks to release CONF_DONE and go into user mode. Figure 13–11 shows how to configure one Cyclone FPGA with one configuration device.

Figure 13–11. Single Device Configuration Circuit



Notes to Figure 13–11:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device. This pull-up resistor is 10 kΩ The EPC16, EPC8, EPC4, and EPC2 devices' OE and nCS pins have internal, user-configurable pull-up resistors. If you use internal pull-up resistors, do not use external pull-up resistors on these pins.

(2) The nINIT_CONF pin is available on EPC16, EPC8, EPC4, and EPC2 devices and has an internal pull-up resistor that is always active. If nINIT_CONF is not used, nCONFIG can be pulled to V_{CC} directly or through a resistor.

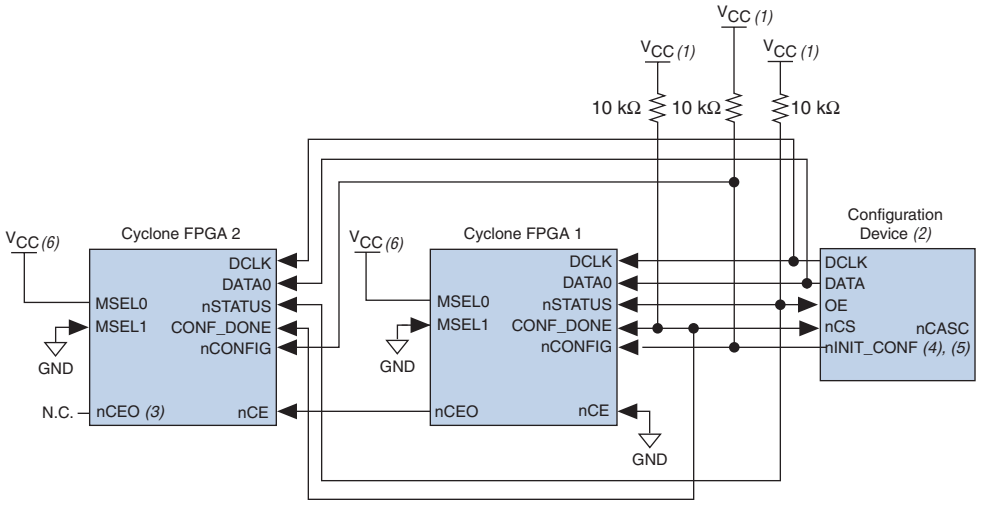
(3) The nCEO pin is left unconnected for the last device in the chain.

(4) Connect MSEL0 to the V_{CC} supply voltage of the I/O bank it resides in.

Configuring Multiple Cyclone FPGAs

You can use a single configuration device to configure multiple Cyclone FPGAs. In this setup, the nCEO pin of the first device is connected to the nCE pin of the second device in the chain. If there are additional devices, connect the nCE pin of the next device to the nCEO pin of the previous device. You should leave the nCEO pin on the last device in the chain unconnected. To configure properly, all of the target device CONF_DONE and nSTATUS pins must be tied together. Figure 13–12 shows an example of configuring multiple Cyclone FPGAs using a single configuration device.

Figure 13–12. Configuring Multiple Cyclone FPGAs with a Single Configuration Device



Notes to Figure 13–12:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device. The EPC16, EPC8, EPC4, and EPC2 devices' OE and nCS pins have internal, user-configurable pull-up resistors. If you use internal pull-up resistors, do not use external pull-up resistors on these pins.
(2) EPC16, EPC8, and EPC4 configuration devices cannot be cascaded.
(3) The nCEO pin is left unconnected for the last device in the chain.
(4) The nINIT_CONF pin is available on EPC16, EPC8, EPC4, and EPC2 devices. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} directly or through a resistor.
(5) The nINIT_CONF pin has an internal pull-up resistor that is always active in EPC16, EPC8, EPC4, and EPC2 devices. These devices do not need an external pull-up resistor on the nINIT_CONF pin.
(6) Connect MSEL0 to the V_{CC} supply voltage of the I/O bank it resides in.

When performing multi-device PS configuration, you must generate the configuration device programming file (**.sof**) from each project. Then you must combine multiple **.sof** files using the Quartus II software through the **Convert Programming Files** dialog box.

After the first Cyclone FPGA completes configuration during multi-device configuration, its nCEO pin activates the second device's nCE pin, prompting the second device to begin configuration. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

In addition, all nSTATUS pins are tied together; therefore, if any device (including the configuration device) detects an error, configuration stops for the entire chain. Also, if the configuration device does not detect CONF_DONE going high at the end of configuration, it resets the chain by

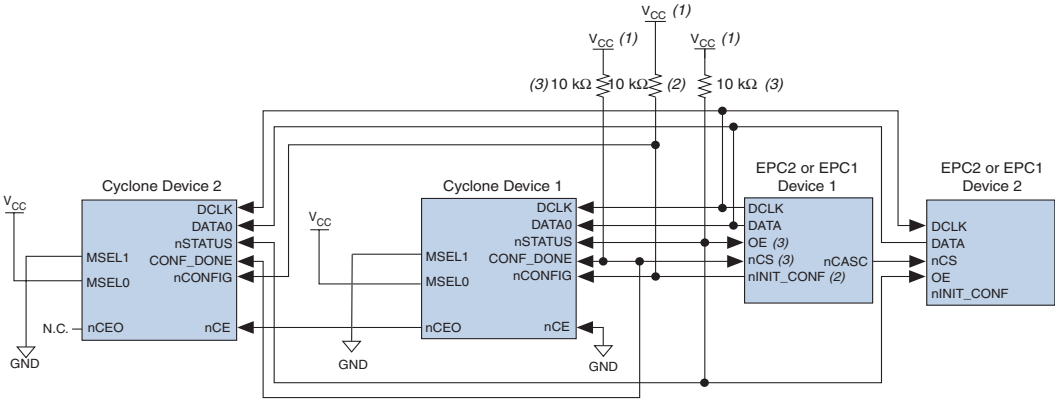
pulsing its OE pin low for a few microseconds. For CONF_DONE to reach a high state, enhanced configuration devices wait for 64 DCLK cycles after the last configuration bit. EPC2 devices wait for 16 DCLK cycles.

If the **Auto-Restart Configuration on Frame Error** option is turned on in the Quartus II software, the Cyclone FPGA releases its nSTATUS pins after a reset time-out period (about 30 µs). When the nSTATUS pins are released and pulled high, the configuration device reconfigures the chain. If the **Auto-Restart Configuration on Frame Error** option is not turned on, the devices drive nSTATUS low until they are reset with a low pulse on nCONFIG.

You can also cascade several EPC2 or EPC1 configuration devices to configure multiple Cyclone FPGAs. When all data from the first configuration device is sent, it drives nCASC low, which in turn drives nCS on the subsequent EPC2 or EPC1 device. Because a configuration device requires less than one clock cycle to activate a subsequent configuration device, the data stream is uninterrupted. You cannot cascade EPC16, EPC8, and EPC4 configuration devices.

Figure 13–13 shows how to configure multiple devices using cascaded EPC2 or EPC1 devices.

Figure 13–13. Multi-Device PS Configuration Using Cascaded EPC2 or EPC1 Devices



Notes to Figure 13–13:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its function is not used. If nINIT_CONF is not used or not available (such as on EPC1 devices), nCONFIG must be pulled to V_{CC} either directly or through a resistor.

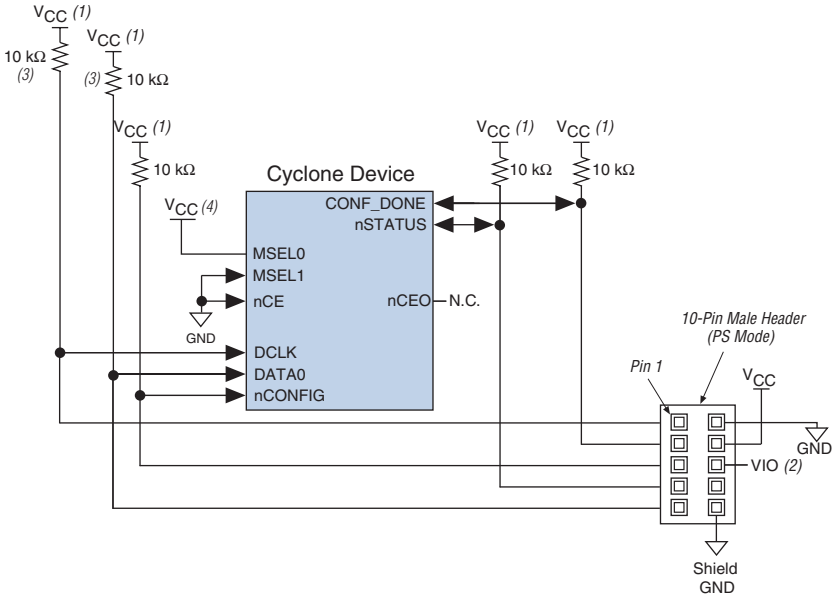
(3) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. External 10-kΩ pull-up resistors should be used. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

PS Configuration Using a Download Cable

Using a download cable in PS configuration, an intelligent host (for example, your PC) transfers data from a storage device (for example, your hard drive) to the Cyclone FPGA through a USB Blaster, ByteBlaster II, MasterBlaster, or ByteBlasterMV cable. To initiate configuration in this scheme, the download cable generates a low-to-high transition on the nCONFIG pin. The programming hardware then sends the configuration data one bit at a time on the device's DATA0 pin. The data is clocked into the target device using DCLK until the CONF_DONE goes high.

When using programming hardware for the Cyclone FPGA, turning on the **Auto-Restart Configuration on Frame Error** option does not affect the configuration cycle because the Quartus II software must restart configuration when an error occurs. Figure 13–14 shows the PS configuration setup for the Cyclone FPGA using a USB Blaster, ByteBlaster II, MasterBlaster, or ByteBlasterMV cable.

Figure 13–14. PS Configuration Circuit with a Download Cable



Notes to Figure 13–14:

(1) You should connect the pull-up resistor to the same supply voltage as the MasterBlaster (VIO pin) or ByteBlasterMV cable.

(2) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. This pin is a no-connect pin for the ByteBlasterMV header.

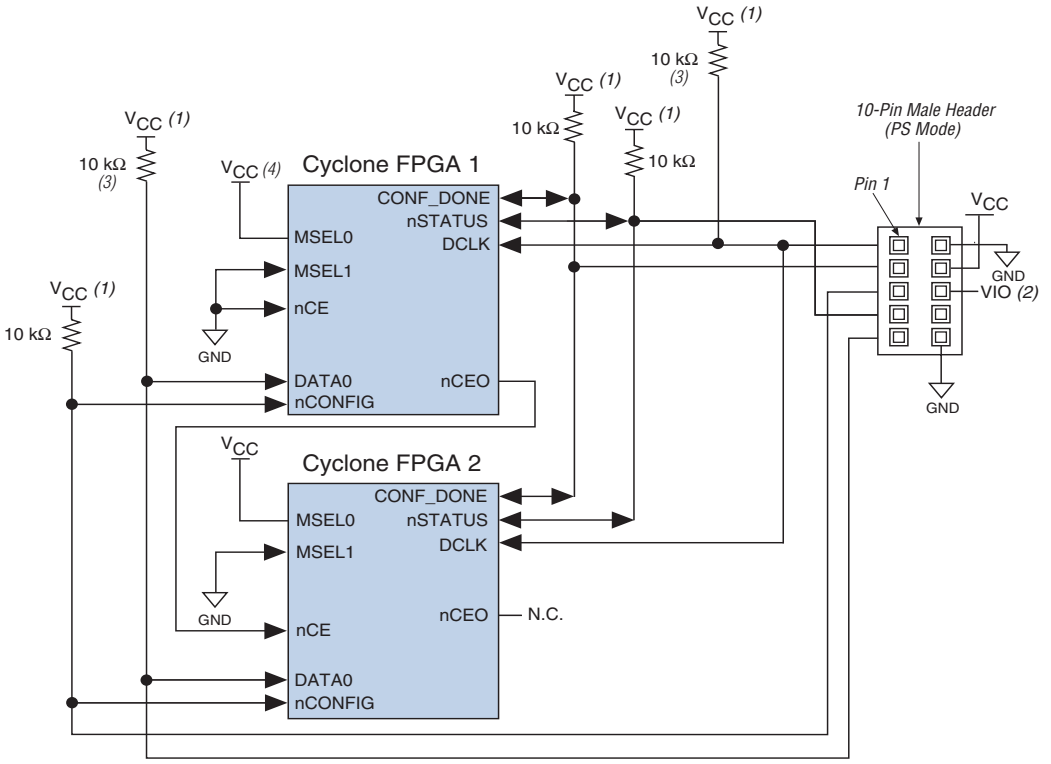
(3) The pull-up resistors on DATA0 and DCLK are only needed if the download cable is the only configuration scheme used on your board. This is to ensure that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, the pull-up resistors on DATA0 and DCLK are not needed.

(4) Connect MSEL0 to the V_{CC} supply voltage of the I/O bank it resides in.

You can use the download cable to configure multiple Cyclone FPGAs by connecting each device's nCEO pin to the subsequent device's nCE pin. All other configuration pins are connected to each device in the chain.

Because all CONF_DONE pins are tied together, all devices in the chain initialize and enter user mode at the same time. In addition, because the nSTATUS pins are tied together, the entire chain halts configuration if any device detects an error. In this situation, the Quartus II software must restart configuration; the **Auto-Restart Configuration on Frame Error** option does not affect the configuration cycle. Figure 13–15 shows how to configure multiple Cyclone FPGAs with a ByteBlaster II, MasterBlaster, or ByteBlasterMV cable.

Figure 13–15. Multi-Device PS Configuration with a Download Cable



Notes to Figure 13–15:

(1) You should connect the pull-up resistor to the same supply voltage as the MasterBlaster (VIO pin) or ByteBlasterMV cable.

(2) V_{IO} is a reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. Refer to the *MasterBlaster Serial/USB Communications Cable User Guide* for this value.

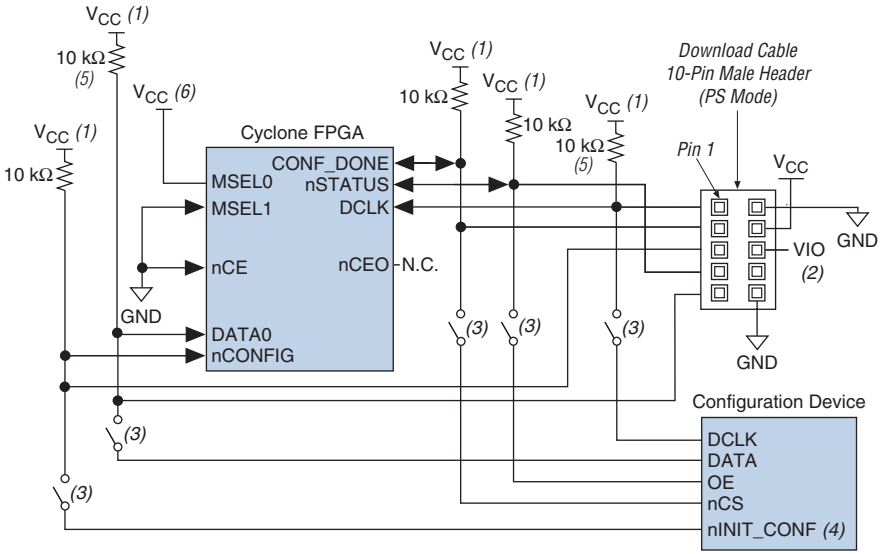
(3) The pull-up resistors on DATA0 and DCLK are only needed if the download cable is the only configuration scheme used on your board. This is to ensure that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, the pull-up resistors on DATA0 and DCLK are not needed.

(4) Connect MSEL0 to the V_{CC} supply voltage of the I/O bank it resides in.

If you are using a ByteBlaster II, MasterBlaster, or ByteBlasterMV cable to configure device(s) on a board that also is populated with configuration devices, you should electrically isolate the configuration devices from the target device(s) and cable. One way to isolate the configuration devices is to add logic, such as a multiplexer, that can select between the configuration devices and the cable. The multiplexer allows bidirectional transfers on the nSTATUS and CONF_DONE signals. Another option is to add switches to the five common signals (CONF_DONE, nSTATUS, DCLK,

nCONFIG, and DATA0) between the cable and the configuration devices. The last option is to remove the configuration devices from the board when configuring with the cable. Figure 13–16 shows a combination of a configuration device and a ByteBlaster II, MasterBlaster, or ByteBlasterMV cable to configure a Cyclone FPGA.

Figure 13–16. Configuring with a Combined PS and Configuration Device Scheme



Notes to Figure 13–16:

(1) You should connect the pull-up resistor to the same supply voltage as the configuration device.

(2) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the target device's V_{CCIO}. This is a no-connect pin for the ByteBlasterMV header.

(3) You should not attempt configuration with a ByteBlaster II, MasterBlaster, or ByteBlasterMV cable while a configuration device is connected to a Cyclone FPGA. Instead, you should either remove the configuration device from its socket when using the download cable or place a switch on the five common signals between the download cable and the configuration device. Remove the ByteBlaster II, MasterBlaster, or ByteBlasterMV cable when configuring with a configuration device.

(4) If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(5) The pull-up resistors on DATA0 and DCLK are only needed if the download cable is the only configuration scheme used on your board. This is to ensure that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, the pull-up resistors on DATA0 and DCLK are not needed.

(6) Connect MSEL0 to the V_{CC} supply voltage of the I/O bank it resides in.

For more information on how to use the ByteBlaster II, MasterBlaster, or ByteBlasterMV cables, see the following documents:

- *ByteBlaster II Download Cable User Guide*
- *ByteBlasterMV Download Cable User Guide*
- *MasterBlaster Serial/USB Communications Cable User Guide*

PS Configuration from a Microprocessor

In PS configuration with a microprocessor, a microprocessor transfers data from a storage device to the target Cyclone FPGA. To initiate configuration in this scheme, the microprocessor must generate a low-to-high transition on the nCONFIG pin and the target device must release nSTATUS. The microprocessor then places the configuration data one bit at a time on the DATA0 pin of the Cyclone FPGA. The least significant bit (LSB) of each data byte must be presented first. Data is clocked continuously into the target device using DCLK until the CONF_DONE signal goes high.

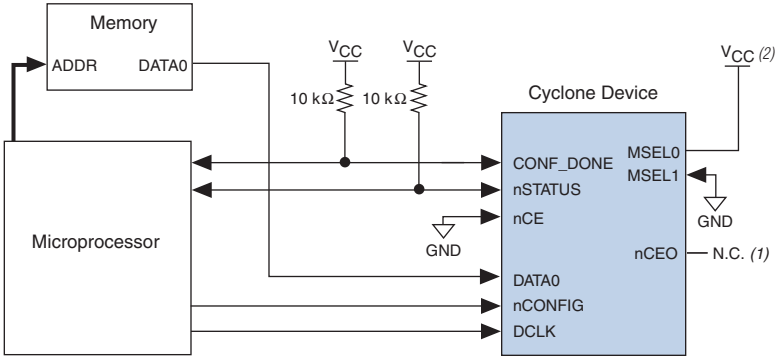
The Cyclone FPGA starts initialization using the internal oscillator after all configuration data is transferred. After initialization, this internal oscillator is turned off. The device's CONF_DONE pin goes high to show successful configuration and the start of initialization. During configuration and initialization and before the device enters user ode the microprocessor must not drive CONF_DONE low. Driving DCLK to the device after configuration does not affect device operation.

Since the PS configuration scheme is a synchronous scheme, the configuration clock speed must be below the specified maximum frequency to ensure successful configuration. Maximum DCLK frequency supported by Cyclone FPGAs is 100 MHz (see Table 13–5 on page 13–30). No maximum DCLK period (i.e., minimum DCLK frequency) exists. You can pause configuration by halting DCLK for an indefinite amount of time.

If the target device detects an error during configuration, it drives its nSTATUS pin low to alert the microprocessor. The microprocessor can then pulse nCONFIG low to restart the configuration process. Alternatively, if the **Auto-Restart Configuration on Frame Error** option is turned on in the Quartus II software, the target device releases nSTATUS after a reset time-out period. After nSTATUS is released, the microprocessor can reconfigure the target device without needing to pulse nCONFIG low.

The microprocessor can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration and initialization. If the microprocessor sends all data, but CONF_DONE and INIT_DONE has not gone high, it must reconfigure the target device. Figure 13–17 shows the circuit for PS configuration with a microprocessor.

Figure 13–17. PS Configuration Circuit with a Microprocessor



Notes to Figure 13–17:
(1) The nCEO pin is left unconnected.
(2) Connect MSEL0 to the V_{CC} supply voltage of the I/O bank it resides in.

Configuring Cyclone FPGAs with the MicroBlaster Software

The MicroBlaster™ software driver allows you to configure Altera FPGAs, including Cyclone FPGAs, through the ByteBlaster II or ByteBlasterMV cable in PS mode. The MicroBlaster software driver supports a Raw Binary File (**.rbf**) programming input file and is targeted for embedded PS configuration. The source code is developed for the Windows NT operating system, although you can customize it to run on other operating systems.

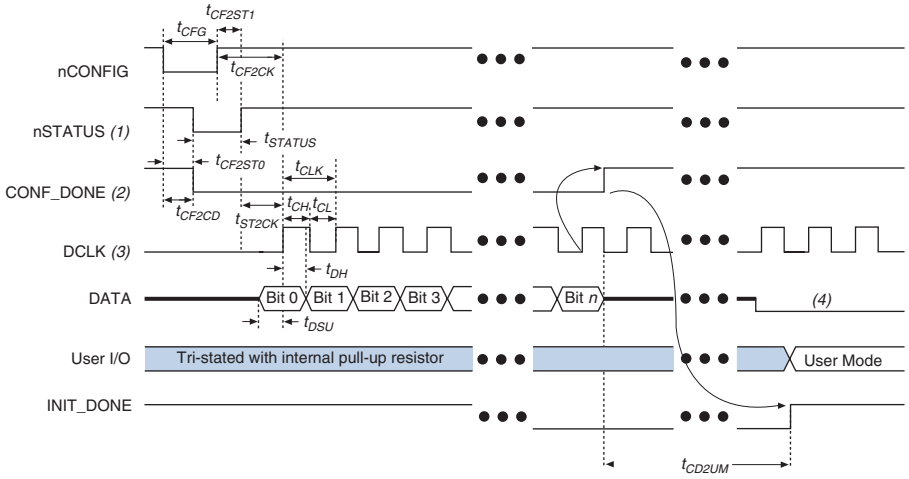
For more information about the MicroBlaster software driver, refer to the *AN 423: Configuring the MicroBlaster Passive Serial Software Driver* and source files on the Altera website at www.altera.com.

Passive Serial Timing

For successful configuration using the PS scheme, several timing parameters such as setup, hold, and maximum clock frequency must be satisfied. The enhanced configuration and EPC2 devices are designed to meet these interface timing specifications. If you use a microprocessor or another intelligent host to control the PS interface, ensure that you meet these timing requirements.

Figure 13–18 shows the PS timing waveform for Cyclone FPGAs.

Figure 13–18. PS Timing Waveform for Cyclone FPGAs



Notes to Figure 13–18:

(1) Upon power-up, the Cyclone FPGA holds nSTATUS low for about 100 ms.
(2) Upon power-up and before configuration, CONF_DONE is low.
(3) In user mode, DCLK should be driven high or low when using the PS configuration scheme. When using the AS configuration scheme, DCLK is a Cyclone output pin and should not be driven externally.
(4) DATA should not be left floating after configuration. It should be driven high or low, whichever is more convenient.

Table 13–5 contains the PS timing information for Cyclone FPGAs.

Table 13–5. PS Timing Parameters for Cyclone Devices Note (1) (Part 1 of 2)

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		800	ns
t_{CF2ST1}	nCONFIG high to nSTATUS high		40 *(4)*	µs
t_{CFG}	nCONFIG low pulse width *(2)*	40		µs
t_{STATUS}	nSTATUS low pulse width	10	40 *(4)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	40		µs
t_{ST2CK}	nSTATUS high to first rising edge on DCLK	1		µs
t_{DSU}	Data setup time before rising edge on DCLK	7		ns
t_{DH}	Data hold time after rising edge on DCLK	0		ns
t_{CH}	DCLK high time	7		ns

Table 13–5. PS Timing Parameters for Cyclone Devices Note (1) *(Part 2 of 2)*

Symbol	Parameter	Min	Max	Units
t_{CL}	DCLK low time	7		ns
t_{CLK}	DCLK period	15		ns
f_{MAX}	DCLK maximum frequency		66	MHz
t_{CD2UM}	CONF_DONE high to user mode *(3)*	6	20	µs

Notes to Table 13–5:

(1) This information is preliminary.
(2) This value applies only if the internal oscillator is selected as the clock source for device initialization. If the clock source is CLKUSR, multiply the clock period by 270 to obtain this value. CLKUSR must be running during this period to reset the device.
(3) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for device initialization. If the clock source is CLKUSR, multiply the clock period by 140 to obtain this value.
(4) You can obtain this value if you do not delay configuration by extending the nSTATUS low-pulse width.

Device configuration options and how to create configuration files are discussed further in the *Software Settings* section in volume 2 of the *Configuration Handbook*.

JTAG-Based Configuration

JTAG has developed a specification for boundary-scan testing. This boundary-scan test (BST) architecture offers the capability to efficiently test components on printed circuit boards (PCBs) with tight lead spacing. The BST architecture can test pin connections without using physical test probes and capture functional data while a device is operating normally. You can also use the JTAG circuitry to shift configuration data into Cyclone FPGAs. The Quartus II software automatically generates **.sof** files that can be used for JTAG configuration.

For more information about JTAG boundary-scan testing, refer to *AN 39: IEEE 1149.1 (JTAG) Boundary-Scan Testing in Altera Devices*.

To use the SignalTap II Embedded Logic Analyzer, you need to connect the JTAG pins of your Cyclone device to a download cableheader on your PCB.

For more information about SignalTap II, refer to the *Design Debugging Using the SignalTap II Embedded Logic Analyzer* chapter in volume 3 of the *Quartus II Handbook*.

Cyclone devices are designed such that JTAG instructions have precedence over any device operating modes. So JTAG configuration can take place without waiting for other configuration to complete (e.g.,

configuration with serial or enhanced configuration devices). If you attempt JTAG configuration in Cyclone FPGAs during non-JTAG configuration, non-JTAG configuration is terminated and JTAG configuration is initiated.

The Cyclone configuration data decompression feature is not supported in JTAG-based configuration.

A device operating in JTAG mode uses four required pins: TDI, TDO, TMS, and TCK. Cyclone FPGAs do not support the optional TRST pin. The three JTAG input pins, TCK, TDI, and TMS, have weak internal pull-up resistors, whose values are approximately 20 to 40 kΩ All user I/O pins are tri-stated during JTAG configuration.

Table 13–6 shows each JTAG pin's function.

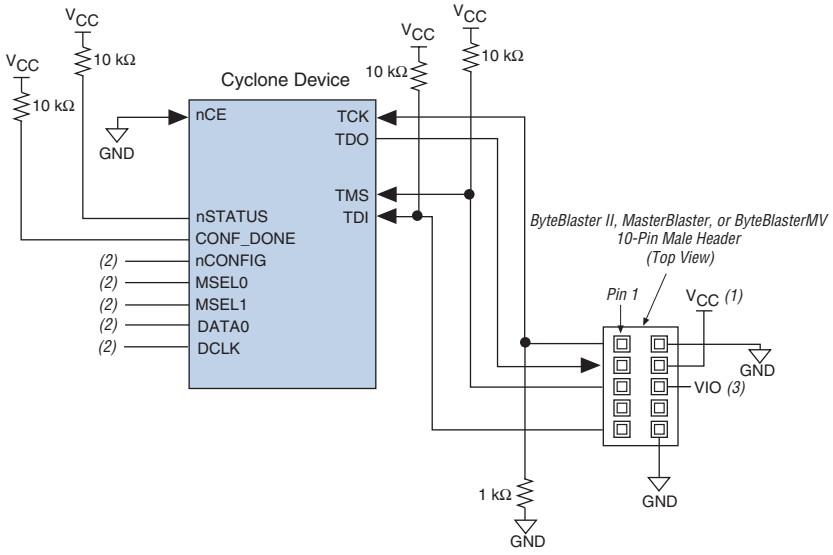
Table 13–6. JTAG Pin Descriptions

Pin	Description	Function
TDI	Test data input	Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}.
TDO	Test data output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by leaving this pin unconnected.
TMS	Test mode select	Input pin that provides the control signal to determine the transitions of the Test Access Port (TAP) controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}.
TCK	Test clock input	The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled, by connecting this pin to GND.

JTAG Configuration Using a Download Cable

During JTAG configuration, data is downloaded to the device on the board through a USB Blaster, ByteBlaster II, ByteBlasterMV, or MasterBlaster download cable. Configuring devices through a cable is similar to programming devices in-system. See Figure 13–19 for pin connection information.

Figure 13–19. JTAG Configuration of Single Cyclone FPGA



Notes to Figure 13–19:

(1) You should connect the pull-up resistor to the same supply voltage as the download cable.

(2) You should connect the nCONFIG, MSEL0, and MSEL1 pins to support a non-JTAG configuration scheme. If you only use JTAG configuration, connect nCONFIG and MSEL0 to V_{CC}, and MSEL1 to ground. Pull DATA0 and DCLK to high or low.

(3) V_{IO} is a reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. Refer to the *MasterBlaster Serial/USB Communications Cable User Guide* for this value. In the ByteBlaster MV, this pin is a no connect. In the USB Blaster and ByteBlaster II, this pin is connected to nCE when it is used for Active Serial programming; otherwise it is a no connect.

(4) nCE must be connected to GND or driven low for successful configuration.

To configure a single device in a JTAG chain, the programming software places all other devices in bypass mode. In bypass mode, devices pass programming data from the TDI pin to the TDO pin through a single bypass register without being affected internally. This scheme enables the programming software to program or verify the target device. Configuration data driven into the device appears on the TDO pin one clock cycle later.

The Quartus II software verifies successful JTAG configuration upon completion. The software checks the state of CONF_DONE through the JTAG port. If CONF_DONE is not high, the Quartus II software indicates that configuration has failed. If CONF_DONE is high, the software indicates that configuration was successful. After the configuration bit stream is transmitted serially via the JTAG TDI port, the TCK port is clocked an additional 134 cycles to perform device initialization.

If V_{CCIO} is tied to 3.3-V, both the I/O pins and the JTAG TDO port drive at 3.3-V levels.

Cyclone FPGAs have dedicated JTAG pins. Not only can you perform JTAG testing on Cyclone FPGAs before and after, but also during configuration. While other device families do not support JTAG testing during configuration, Cyclone FPGAs support the BYPASS, IDCODE, and SAMPLE instructions during configuration without interrupting configuration. All other JTAG instructions may only be issued by first interrupting configuration and reprogramming I/O pins using the CONFIG_IO instruction.

The CONFIG_IO instruction allows I/O buffers to be configured via the JTAG port, and when issued, interrupts configuration. This instruction allows you to perform board-level testing prior to configuring the Cyclone FPGA or waiting for a configuration device to complete configuration. Once configuration has been interrupted and JTAG testing is complete, the part must be reconfigured via JTAG (PULSE_CONFIG instruction) or by pulsing nCONFIG low.

The chip-wide reset and output enable pins on Cyclone FPGAs do not affect JTAG boundary-scan or programming operations. Toggling these pins does not affect JTAG operations (other than the usual boundary-scan operation).

When designing a board for JTAG configuration of Cyclone FPGAs, you should consider the dedicated configuration pins. Table 13–7 shows how you should connect these pins during JTAG configuration.

Table 13–7. Dedicated Configuration Pin Connections During JTAG Configuration

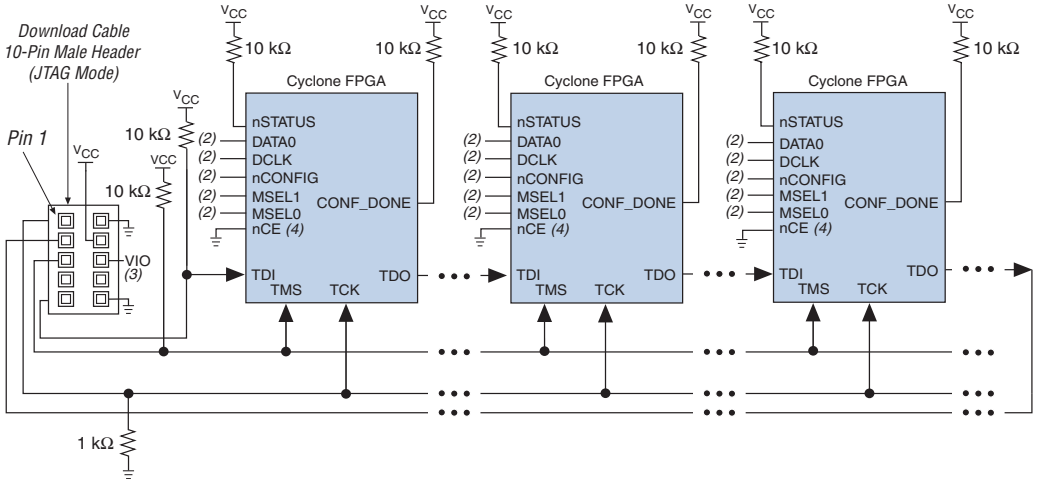
Signal	Description
nCE	Drive all Cyclone devices in the chain low by connecting nCE to ground, pulling it down via a resistor, or driving it low by some control circuitry. For devices in a multi-device PS and AS configuration chains, connect the nCE pins to ground during JTAG configuration or configure them via JTAG in the same order as the configuration chain.
nCEO	For all Cyclone devices in a chain, the nCEO pin can be left floating or connected to the nCE pin of the next device. See nCE description above.
nSTATUS	Pulled to V_{CC} through a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, pull up each nSTATUS pin to V_{CC} individually.
CONF_DONE	Pulled to V_{CC} through a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, pull up each CONF_DONE pin to V_{CC} individually. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.
nCONFIG	Driven high by connecting to V_{CC}, pulling up through a resistor, or driving it high by some control circuitry.
MSEL0, MSEL1	Do not leave these pins floating. These pins support whichever non-JTAG configuration is used in production. If only JTAG configuration is used, you should tie these pins to ground.
DCLK	Do not leave these pins floating. Drive low or high, whichever is more convenient.
DATA0	Do not leave these pins floating. Drive low or high, whichever is more convenient.

JTAG Configuration of Multiple Devices

When programming a JTAG device chain, one JTAG-compatible header, such as the ByteBlaster II header, is connected to several devices. The number of devices in the JTAG chain is limited only by the drive capacity of the download cable. However, when four or more devices are connected in a JTAG chain, Altera recommends buffering the TCK, TDI, and TMS pins with an on-board buffer.

JTAG-chain device configuration is ideal when the system contains multiple devices, or when testing your system using JTAG BST circuitry. Figure 13–20 shows multi-device JTAG configuration.

Figure 13–20. Multi-Device JTAG Configuration *Note (1)*



Notes to Figure 13–20:

(1) Cyclone, Stratix, Stratix GX, APEX™ II, APEX 20K, Mercury™, ACEX® 1K, and FLEX® 10K devices can be placed within the same JTAG chain for device programming and configuration.

(2) Connect the nCONFIG, MSEL0, and MSEL1 pins to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect nCONFIG to V_{CC}, and MSEL0 and MSEL1 to ground. Pull DATA0 and DCLK to either high or low.

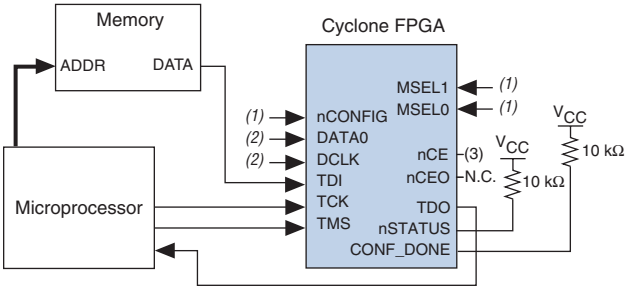
(3) V_{IO} is a reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. Refer to the *MasterBlaster Serial/USB Communications Cable User Guide* for this value. In the ByteBlaster MV, this pin is a no connect. In the USB Blaster and ByteBlaster, this pin is connected to nCE when it is used for Active Serial programming; otherwise it is a no connect.

(4) nCE must be connected to GND or driven low for successful configuration.

Connect the nCE pin to ground or drive it low during JTAG configuration. In multi-device PS and AS configuration chains, connect the first device's nCE pin to ground and connect the nCEO pin to the nCE pin of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, it's nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. Therefore, if these devices are also in a JTAG chain, you should make sure the nCE pins are connected to ground during JTAG configuration or that the devices are configured via JTAG in the same order as the configuration chain. As long as the devices are configured in the same order as the multi-device configuration chain, the nCEO pin of the previous device drives the nCE pin of the next device low when it has successfully been configured.

Figure 13–21 shows the JTAG configuration of a Cyclone FPGA with a microprocessor.

Figure 13–21. JTAG Configuration of Cyclone FPGAs with a Microprocessor



Notes to Figure 13–21:

(1) Connect the nCONFIG, MSEL1, and MSEL0 pins to support a non-JTAG configuration scheme. If your design only uses JTAG configuration, connect the nCONFIG pin to V_{CC} and the MSEL1 and MSEL0 pins to ground.

(2) Pull DATA0 and DCLK to either high or low.

(3) nCE must be connected to GND or driver low for succesful JTAG configuration.

For more information about JTAG programming in an embedded environment, refer to *AN 122: Using JamSTAPL for ISP &ICR via an Embedded Processor.*

Configuring Cyclone FPGAs with JRunner

JRunner is a software driver that allows you to configure Altera FPGAs, including Cyclone FPGAs, through the ByteBlaster II or ByteBlasterMV cables in JTAG mode. The programming input file supported is in **.rbf** format. JRunner also requires a Chain Description File (**.cdf**) generated by the Quartus II software. JRunner is targeted for embedded JTAG configuration. The source code has been developed for the Windows NT operating system (OS). You can customize the code to make it run on other platforms. For more information on the JRunner software driver, see JRunner Software Driver: An Embedded Solution to the JTAG Configuration and the source files on the Altera website.

Jam STAPL

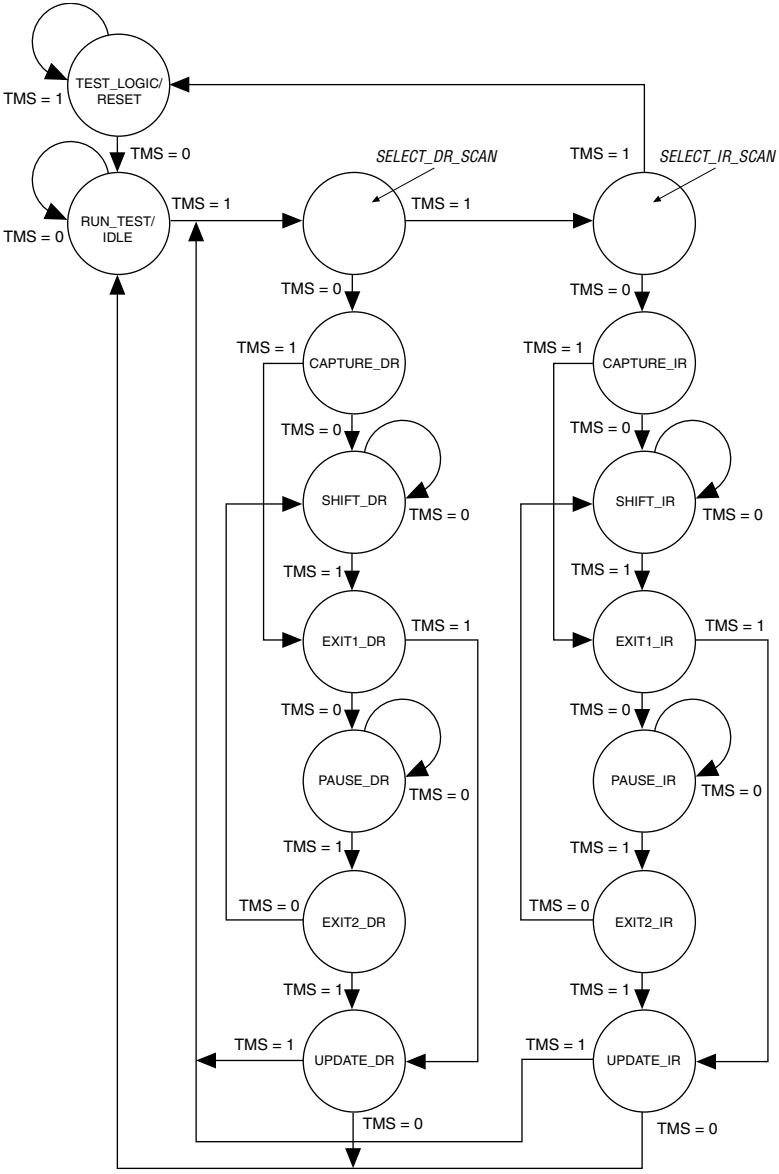
Jam STAPL, JEDEC standard JESD-71, is a standard file format for in-system programmability (ISP) purposes. Jam STAPL supports programming or configuration of programmable devices and testing of electronic systems, using the IEEE 1149.1 JTAG interface. Jam STAPL is a freely licensed open standard.

Both JTAG connection methods should include space for the MasterBlaster or ByteBlasterMV header connection. The header is useful during prototyping because it allows you to verify or modify the Cyclone FPGA's contents. During production, you can remove the header to save cost.

Program Flow

The Jam Player provides an interface for manipulating the IEEE Std. 1149.1 JTAG TAP state machine. The TAP controller is a 16-state, state machine that is clocked on the rising edge of `TCK`, and uses the `TMS` pin to control JTAG operation in a device. Figure 13–22 shows the flow of an IEEE Std. 1149.1 TAP controller state machine.

Figure 13–22. JTAG TAP Controller State Machine



While the Jam Player provides a driver that manipulates the TAP controller, the Jam Byte-Code File (**.jbc**) provides the high-level intelligence needed to program a given device. All Jam instructions that

send JTAG data to the device involve moving the TAP controller through either the data register leg or the instruction register leg of the state machine. For example, loading a JTAG instruction involves moving the TAP controller to the SHIFT_IR state and shifting the instruction into the instruction register through the TDI pin. Next, the TAP controller is moved to the RUN_TEST/IDLE state where a delay is implemented to allow the instruction time to be latched. This process is identical for data register scans, except that the data register leg of the state machine is traversed.

The high-level Jam instructions are the DRSCAN instruction for scanning the JTAG data register, the IRSCAN instruction for scanning the instruction register, and the WAIT command that causes the state machine to sit idle for a specified period of time. Each leg of the TAP controller is scanned repeatedly, according to instructions in the **.jbc** file, until all of the target devices are programmed.

Figure 13–23 shows the functional behavior of the Jam Player when it parses the **.jbc** file. When the Jam Player encounters a DRSCAN, IRSCAN, or WAIT instruction, it generates the proper data on TCK, TMS, and TDI to complete the instruction. The flow diagram shows branches for the DRSCAN, IRSCAN, and WAIT instructions. Although the Jam Player supports other instructions, they are omitted from the flow diagram for simplicity.

Figure 13–23. Jam Player Flow Diagram (Part 1 of 2)

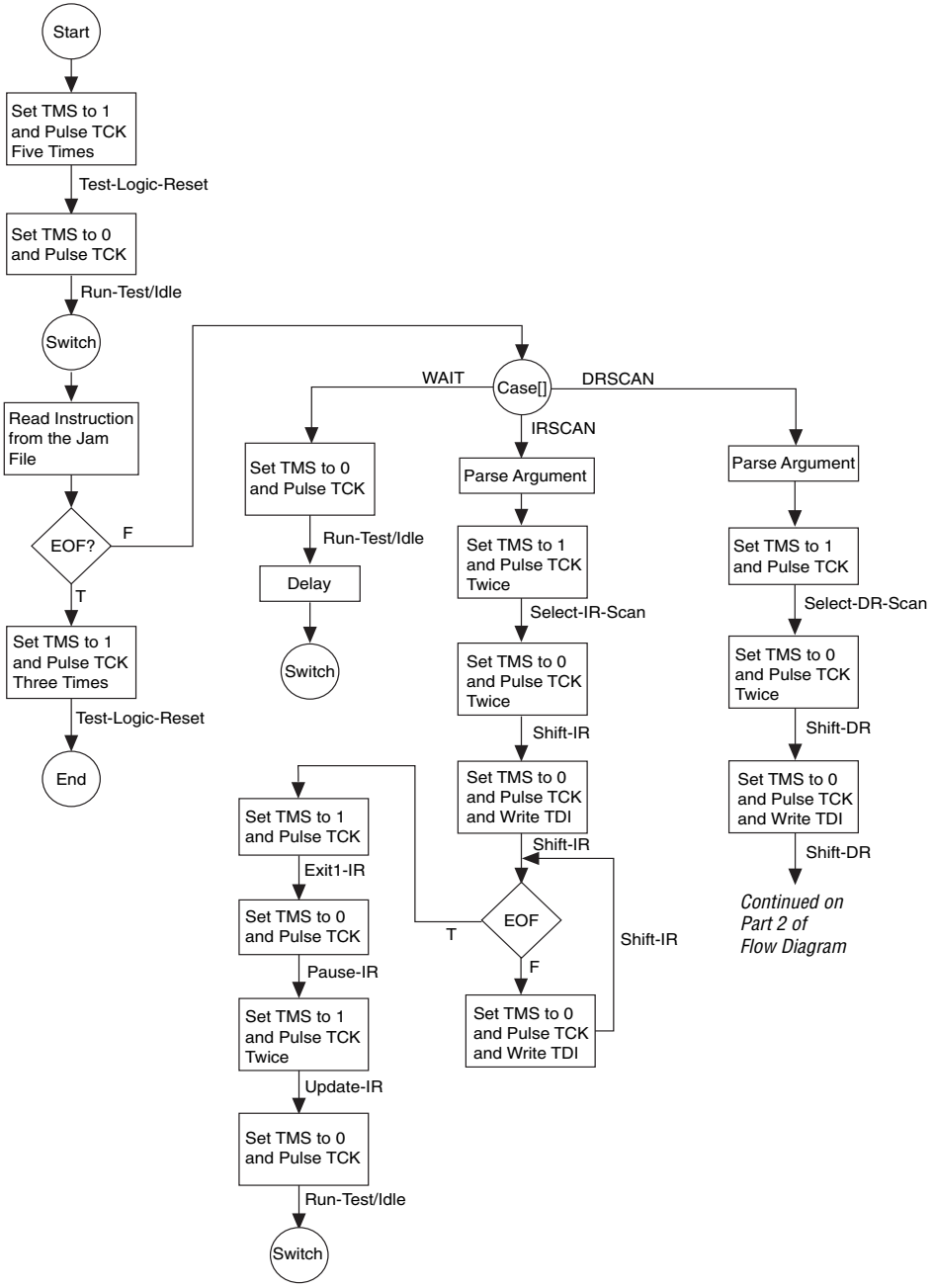
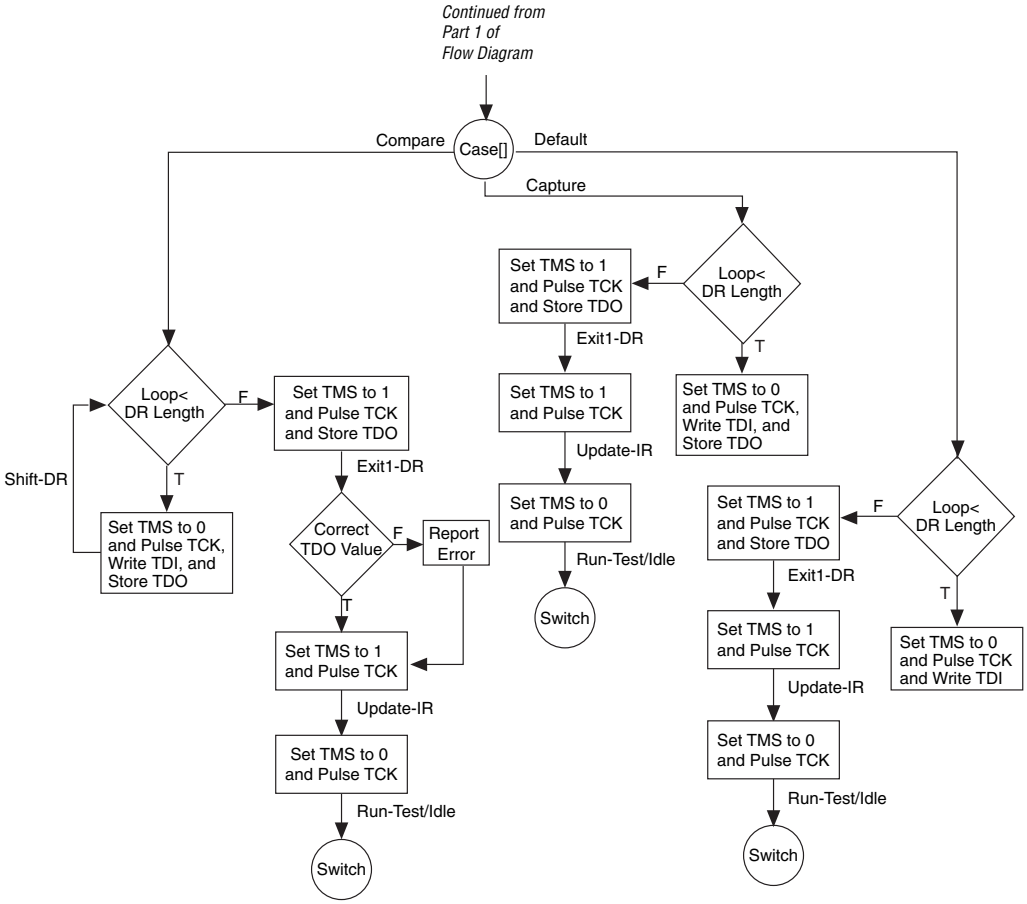


Figure 13–24. Jam Player Flow Diagram (Part 2 of 2)



Execution of a Jam program starts at the beginning of the program. The program flow is controlled using `GOTO`, `CALL/RETURN`, and `FOR/NEXT` structures. The `GOTO` and `CALL` statements refer to labels that are symbolic names for program statements located elsewhere in the Jam program. The language itself enforces almost no constraints on the organizational structure or control flow of a program.

The Jam language does not support linking multiple Jam programs together or including the contents of another file into a Jam program.

Jam Instructions

Each Jam statement begins with one of the instruction names listed in Table 13–8. The instruction names, including the names of the optional instructions, are reserved keywords that you cannot use as variable or label identifiers in a Jam program.

Table 13–8. Instruction Names

BOOLEAN	INTEGER	PREIR
CALL	IRSCAN	PRINT
CRC	IRSTOP	PUSH
DRSCAN	LET	RETURN
DRSTOP	NEXT	STATE
EXIT	NOTE	WAIT
EXPORT	POP	VECTOR *(1)*
FOR	POSTDR	VMAP *(1)*
GOTO	POSTIR	–
IF	PREDR	–

Note to Table 13–8:
(1) This instruction name is an optional language extension.

Table 13–9 shows the state names that are reserved keywords in the Jam language. These keywords correspond to the state names specified in the IEEE Std. 1149.1 JTAG specification.

Table 13–9. Reserved Keywords (Part 1 of 2)

IEEE Std. 1149.1 JTAG State Names	Jam Reserved State Names
Test-Logic-Reset	RESET
Run-Test-Idle	IDLE
Select-DR-Scan	DRSELECT
Capture-DR	DRCAPTURE
Shift-DR	DRSHIFT
Exit1-DR	DREXIT1
Pause-DR	DRPAUSE
Exit2-DR	DREXIT2
Update-DR	DRUPDATE
Select-IR-Scan	IRSELECT
Capture-IR	IRCAPTURE

Table 13–9. Reserved Keywords (Part 2 of 2)	
IEEE Std. 1149.1 JTAG State Names	**Jam Reserved State Names**
Shift-IR	IRSHIFT
Exit1-IR	IREXIT1
Pause-IR	IRPAUSE
Exit2-IR	IREXIT2
Update-IR	IRUPDATE

Example Jam File that Reads the IDCODE

The following illustrates the flexibility and utility of the Jam STAPL. The example code reads the IDCODE out of a single device in a JTAG chain.

The array variable, I_IDCODE, is initialized with the IDCODE instruction bits ordered the LSB first (on the left) to most significant bit (MSB) (on the right). This order is important because the array field in the IRSCAN instruction is always interpreted and sent, MSB to LSB.

Example Jam File Reading IDCODE

```
BOOLEAN read_data[32];
BOOLEAN I_IDCODE[10] = BIN 1001101000; 'assumed
BOOLEAN ONES_DATA[32] = HEX FFFFFFFF;
INTEGER i;
'Set up stop state for IRSCAN
IRSTOP IRPAUSE;
'Initialize device
STATE RESET;
IRSCAN 10, I_IDCODE[0..9]; 'LOAD IDCODE INSTRUCTION
STATE IDLE;
WAIT 5 USEC, 3 CYCLES;
DRSCAN 32, ONES_DATA[0..31], CAPTURE read_data[0..31];
'CAPTURE IDCODE
PRINT "IDCODE:";
FOR i=0 to 31;
PRINT read_data[i];
NEXT i;
EXIT 0;
```

Combining Configuration Schemes

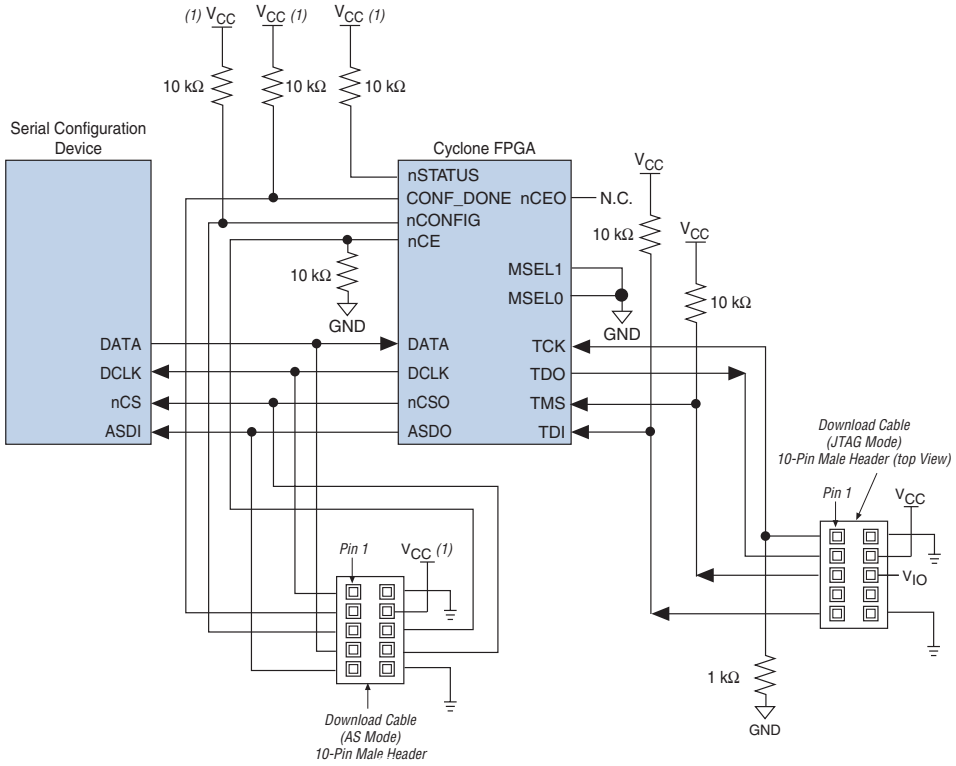
This section shows you how to configure Cyclone FPGAs using multiple configuration schemes on the same board.

Active Serial and JTAG

You can combine the AS configuration scheme with JTAG-based configuration. Set the `MSEL[1..0]` pins to `00` in this setup, as shown in Figure 13–25. This setup uses two 10-pin download cable headers on the board. The first header programs the serial configuration device in-system via the AS programming interface, and the second header configures the Cyclone FPGA directly via the JTAG interface.

If you try configuring the device using both schemes simultaneously, JTAG configuration takes precedence and AS configuration is terminated.

Figure 13–25. Combining AS and JTAG Configuration



Note to Figure 13–25:
(1) Connect these pull-up resistors to 3.3 V.

Device Configuration Pins

Tables 13–10 through 13–12 describe the connections and functionality of all the configuration related pins on the Cyclone device. Table 13–10 describes the dedicated configuration pins. These pins are required to be connected properly on your board for successful configuration. Some of these pins may not be required for your configuration schemes.

Table 13–10. Dedicated Cyclone Device Configuration Pins (Part 1 of 3)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
MSEL1 MSEL0	–	All	Input	Two-bit configuration input that set the Cyclone device configuration scheme (see Table 13–2). Use these pins to select the Cyclone configuration schemes for the appropriate connections. These pins must remain at a valid state during power-up before nCONFIG is pulled low to initiate a reconfiguration and during configuration. This pin uses Schmitt trigger input buffers.
nCONFIG	–	All	Input	Configuration control input. Pulling this pin low during user-mode causes the FPGA to lose its configuration data, enter a reset state, and tri-state all I/O pins. Returning this pin to a logic high initiates a reconfiguration. If the configuration scheme uses an enhanced configuration device or EPC2 device, the nCONFIG pin can be tied directly to V_{CC} or to the configuration device's nINIT_CONF pin. This pin uses Schmitt trigger input buffers

Table 13–10. Dedicated Cyclone Device Configuration Pins (Part 2 of 3)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nSTATUS	–	All	Bidirectional open-drain	The device drives nSTATUS low immediately after power-up and releases it within 5 µs. (When using a configuration device, the configuration device holds nSTATUS low for up to 200 ms.) Status output. If an error occurs during configuration, nSTATUS is pulled low by the target device. Status input. If an external source drives the nSTATUS pin low during configuration or initialization, the target device enters an error state. Driving nSTATUS low after configuration and initialization does not affect the configured device. If the design uses a configuration device, driving nSTATUS low causes the configuration device to attempt to configure the FPGA, but since the FPGA ignores transitions on nSTATUS in user-mode, the FPGA does not reconfigure. To initiate a reconfiguration, nCONFIG must be pulled low. The OE and nCS pins in the enhanced configuration devices and EPC2 devices have optional internal programmable pull-up resistors. If the design uses internal pull-up resistors, do not use external 10-kΩ pull-up resistors on these pins. This pin uses Schmitt trigger input buffers
CONF_DONE	–	All	Bidirectional open-drain	Status output. The target device drives the CONF_DONE pin low before and during configuration. Once all configuration data is received without error and the initialization clock cycle starts, the target device releases CONF_DONE. Status input. After all data is received and CONF_DONE goes high, the target device initializes and enters user mode. Driving CONF_DONE low after configuration and initialization does not affect the configured device. The OE and nCS pins in the enhanced configuration devices and EPC2 devices have optional internal programmable pull-up resistors. If the design uses internal pull-up resistors, do not use external 10-kΩ pull-up resistors on these pins. This pin uses Schmitt trigger input buffers

Table 13–10. Dedicated Cyclone Device Configuration Pins (Part 3 of 3)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
DCLK	–	PS AS	Input (PS) Output (AS)	In PS configuration, the clock input clocks data from an external source into the target device. Data is latched into the FPGA on the rising edge of DCLK. In AS configuration, DCLK is an output from the Cyclone FPGA that provides timing for the configuration interface. After configuration, the logic levels on this pin do not affect the Cyclone FPGA. This pin uses Schmitt trigger input buffers
ASDO	I/O in PS mode, N/A in AS mode	AS	Output	Control signal from the Cyclone FPGA to the serial configuration device in AS mode used to read out configuration data.
nCSO	I/O in PS mode, N/A in AS mode	AS	Output	Output control signal from the Cyclone FPGA to the serial configuration device in AS mode that enables the configuration device.
nCE	–	All	Input	Active-low chip enable. The nCE pin activates the device with a low signal to allow configuration. The nCE pin must be held low during configuration, initialization, and user mode. In single device configuration, tie the nCE pin low. In multi-device configuration, the first device's nCE pin is tied low while its nCEO pin is connected to nCE of the next device in the chain. Hold the nCE pin low for programming the FPGA via JTAG. This pin uses Schmitt trigger input buffers
nCEO	–	All	Output	Output that drives low when device configuration is complete. In single device configuration, this pin is left floating. In multi-device configuration, this pin feeds the next device's nCE pin. The nCEO of the last device in the chain is left floating.
DATA0	–	All	Input	Data input. In serial configuration mode, bit-wide configuration data is presented to the target device on the DATA0 pin. Toggling DATA0 after configuration does not affect the configured device. This pin uses Schmitt trigger input buffers

Table 13–11 describes the optional configuration pins. If these optional configuration pins are not enabled in the Quartus II software, they are available as general-purpose user I/O pins. Therefore during configuration, these pins function as user I/O pins and are tri-stated with weak pull-ups.

Table 13–11. Optional Cyclone Device Configuration Pins

Pin Name	User Mode	Pin Type	Description
CLKUSR	N/A if option is on, I/O if option is off	Input	Optional user-supplied clock input. Synchronizes the initialization of one or more devices. This pin is enabled by turning on the **Enable user-supplied start-up clock (CLKUSR)** option in the Quartus II software.
INIT_DONE	N/A if option is on, I/O if option is off	Output open-drain	Status pin. Can be used to indicate when the device has initialized and is in user mode. The INIT_DONE pin must be pulled to V_{CC} with a 10-kΩ resistor. The INIT_DONE pin drives low during configuration. Before and after configuration, the INIT_DONE pin is released and is pulled to V_{CC} by an external pull-up resistor. Because INIT_DONE is tri-stated before configuration, it is pulled high by the external pull-up resistor. Thus, the monitoring circuitry must be able to detect a low-to-high transition. This pin is enabled by turning on the **Enable INIT_DONE output** option in the Quartus II software.
DEV_OE	N/A if the option is on, I/O if the option is off.	Input	Optional pin that allows the user to override all tri-states on the device. When this pin is driven low, all I/O pins are tri-stated; when this pin is driven high, all I/O pins behave as programmed. This pin is enabled by turning on the **Enable device-wide output enable (DEV_OE)** option in the Quartus II software.
DEV_CLRn	N/A if the option is on, I/O if the option is off.	Input	Optional pin that allows you to override all clears on all device registers. When this pin is driven low, all registers are cleared; when this pin is driven high, all registers behave as programmed. This pin is enabled by turning on the **Enable device-wide reset (DEV_CLRn)** option in the Quartus II software.

Table 13–12 describes the dedicated JTAG pins. JTAG pins must be kept stable before and during configuration to prevent accidental loading of JTAG instructions.

Table 13–12. Dedicated JTAG Pins

Pin Name	User Mode	Pin Type	Description
TDI	N/A	Input	Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}. This pin uses Schmitt trigger input buffers
TDO	N/A	Output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by leaving this pin unconnected.
TMS	N/A	Input	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}. This pin uses Schmitt trigger input buffers
TCK	N/A	Input	The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to ground. This pin uses Schmitt trigger input buffers

Referenced Documents

This chapter references the following documents:

- *AN 39: IEEE 1149.1 (JTAG) Boundary-Scan Testing in Altera Devices*
- *AN 418: SRunner: An Embedded Solution for Serial Configuration Device Programming*
- *AN 423: Configuring the MicroBlaster Passive Serial Software Driver*
- *ByteBlaster II Download Cable User Guide*
- *ByteBlasterMV Download Cable User Guide*
- *Cyclone FPGA Family Data Sheet* section of the *Cyclone Device Handbook*
- *DC and Switching Characteristics* chapter in the *Cyclone Device Handbook*
- *Design Debugging Using the SignalTap II Embedded Logic Analyzer* chapter in volume 3 of the *Quartus II Handbook*
- *MasterBlaster Serial/USB Communications Cable User Guide*
- *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet*
- *Software Settings* section in volume 2 of the *Configuration Handbook*

Document Revision History

Table 13–13 shows the revision history for this chapter.

Table 13–13. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
May 2008 v1.8	Minor textual and style changes. Added "Referenced Documents" section.	—
January 2007 v1.7	• Added document revision history. • Removed a note from Table 13–2. • Updated Figure 13–1. • Updated Table 13–3. • Updated feetpara note in "Active Serial Configuration (Serial Configuration Devices)" section. • Updated feetpara note on page 13–18. • Updated *Note (2)* in Figure 13–11. • Updated *Note (4)* in Figure 13–12. • Updated *Note (2)* in Figure 13–19.	—
July 2006 v1.6	Updated Figure 13–19.	—
August 2005 v1.5	• Updated tables. • Minor text updates.	—
March 2005 v1.4	• Updated Figure 13–1. • Updated Figure 13–10.	—
February 2005 v1.3	Updated Figure 13–13.	—
August 2004 v1.2	• Deleted sections: Programming Configuration Devices, Connecting the JTAG Chain, Passive Serial and JTAG, Device Options, Device Configuration Files, Configuration Reliability, and Board Layout Tips. • Deleted figures: Embedded System Block Diagram, Combining PS & JTAG Configuration, Configuration Options Dialog Box. • Deleted table: Cyclone Configuration Option Bits. • Added: USB Blaster to cable list; new Figure 13–13; text on pages 13-14, 13-29, and 13-30, and information to Table 13–6. • Changes to Figures 13–14 to 13–16, 13–19, 13–20, 13–25; numbers changed in EP1C4 row of Table 13–3. • Added extensive descriptions of configuration methods under the "Configuring Multiple Devices with the Same Data" section.	—
July 2003 v1.1	Updated **.rbf** sizes. Minor updates throughout the document.	—
May 2003 v1.0	Added document to *Cyclone Device Handbook*.	—

9. Configuration, Design Security, and Remote System Upgrades in Arria II Devices

AIIGX51009-4.1

This chapter describes the supported configuration schemes for Arria® II devices, instructions for executing the required configuration schemes, and the necessary option pin settings. This chapter also reviews the different ways you can configure your device and explains the design security and remote system upgrade features for Arria II devices.

Arria II devices use SRAM cells to store configuration data. Because SRAM memory is volatile, you must download configuration data to the Arria II device each time the device powers up. All configuration schemes use either an external controller (for example, a MAX® II device or microprocessor), a configuration device, or a download cable.

This chapter includes the following sections:

- "Configuration Features"
- "Power-On Reset Circuit and Configuration Pins Power Supply" on page 9–4
- "Configuration Process" on page 9–7
- "Configuration Schemes" on page 9–9
- "Fast Passive Parallel Configuration" on page 9–11
- "AS and Fast AS Configuration (Serial Configuration Devices)" on page 9–19
- "PS Configuration" on page 9–27
- "JTAG Configuration" on page 9–34
- "Device Configuration Pins" on page 9–40
- "Configuration Data Decompression" on page 9–47
- "Remote System Upgrades" on page 9–49
- "Remote System Upgrade Mode" on page 9–53
- "Dedicated Remote System Upgrade Circuitry" on page 9–56
- "Quartus II Software Support" on page 9–61
- "Design Security" on page 9–62

© 2011 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

Configuration Devices

Altera® serial configuration devices support single- and multi-device configuration solutions for Arria II devices. Arria II GX devices use the active serial (AS) configuration scheme while Arria II GZ devices use the fast AS configuration scheme. Serial configuration devices offer a low-cost, low pin-count configuration solution.

For more information about serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Datasheet* in volume 2 of the *Configuration Handbook*.

All minimum timing information stated in this chapter covers the entire Arria II device family. Some devices may work at less than the minimum timing stated in this chapter due to process variations.

Configuration Features

Arria II devices offer decompression, design security, and remote system upgrade features. Arria II devices can receive a compressed configuration bitstream and decompress this data in real-time, reducing storage requirements and configuration time. Design security using configuration bitstream encryption protects your designs. You can make real-time system upgrades from remote locations of your Arria II designs with the remote system upgrade feature.

Table 9–1 lists the configuration features you can use in each configuration scheme for Arria II GX devices.

Table 9–1. Configuration Features for Arria II GX Devices

Configuration Scheme	Configuration Method	Decompression	Design Security	Remote System Upgrade
FPP	MAX II device or a microprocessor with flash memory	✓ *(1)*	✓ *(1)*	—
AS	Serial configuration device	✓	✓	✓ *(2)*
PS	MAX II device or a microprocessor with flash memory	✓	✓	—
	Download cable	✓	✓	—
JTAG	MAX II device or a microprocessor with flash memory	—	—	—
	Download cable	—	—	—

Notes to Table 9–1:

(1) In these modes, the host system must send a DCLK that is x4 the data rate.

(2) Remote system upgrade is only available in the AS configuration scheme. Local update mode is not supported in the AS configuration scheme.

Table 9–2 lists the configuration features you can use in each configuration scheme for Arria II GZ devices.

Table 9–2. Configuration Features for Arria II GZ Devices

Configuration Scheme	Configuration Method	Decompression	Design Security	Remote System Upgrade
FPP	MAX II device or a microprocessor with flash memory	✓ *(1)*	✓ *(1)*	—
Fast AS	Serial configuration device	✓	✓	✓ *(2)*
PS	MAX II device or a microprocessor with flash memory	✓	✓	—
	Download cable	✓	✓	—
JTAG	MAX II device or a microprocessor with flash memory	—	—	—
	Download cable	—	—	—

Notes to Table 9–2:

(1) In these modes, the host system must send a DCLK that is x4 the data rate.

(2) Remote system upgrade is only available in the fast AS configuration scheme. Local update mode is not supported in the fast AS configuration scheme.

Refer to the following for the configuration features supported in Arria II devices:

- For more information about the configuration data decompression feature, refer to "Configuration Data Decompression" on page 9–47.
- For more information about the remote system upgrade feature, refer to "Remote System Upgrades" on page 9–49.
- For more information about the design security feature, refer to the "Design Security" on page 9–62.
- For more information about the parallel flash loader (PFL), refer to *Parallel Flash Loader Megafunction User Guide*.

If your system already contains a common flash interface (CFI) flash memory device, you can also use it for the Arria II device configuration storage. The PFL feature in MAX II devices provides an efficient method to program CFI flash memory devices through the JTAG interface and provides the logic to control configuration from the flash memory device to the Arria II device. Both passive serial (PS) and fast passive parallel (FPP) configuration modes are supported using this PFL feature.

For more information about programming Altera serial configuration devices, refer to "Programming Serial Configuration Devices" on page 9–23.

Power-On Reset Circuit and Configuration Pins Power Supply

The following sections describe the power-on reset (POR) circuit and the power supply for the configuration pins.

Power-On Reset Circuit

The POR circuit keeps the entire system in reset mode until the power supply voltage levels have stabilized on power-up. After power-up, the device does not release nSTATUS until the voltage levels are above the POR trip point of the device. Table 9–3 lists the voltages required for power-up in Arria II devices.

Table 9–3. Required Voltages for Arria II Devices

Devices	Voltages
Arria II GX	V_{CCCB}, V_{CCA_PLL}, V_{CC}, V_{CCPD}, and V_{CCIO} for I/O banks 3C or 8C
Arria II GZ	V_{CC}, V_{CCAUX}, V_{CCCB}, V_{CCPGM}, and V_{CCPD}

On power down for Arria II GX devices, brown-out occurs if V_{CC} ramps down below the POR trip point and any of the V_{CC}, V_{CCPD}, or V_{CCIO} voltages for I/O banks 3C or 8C drops below the threshold. On power down for Arria II GZ devices, brown-out occurs if the V_{CC}, V_{CCAUX}, V_{CCCB}, V_{CCPGM}, or V_{CCPD} voltages drops below the threshold voltage.

In Arria II devices, you can select between a fast POR time or a standard POR time. For Arria II GX devices, selection depends on the MSEL pin settings. For Arria II GZ devices, selection depends on the PORSEL input pin. PORSEL = L is set as standard POR time. PORSEL = H is set as fast POR time. Fast POR time is **4 ms < T_{POR} < 12 ms** for a fast configuration time. Standard POR time is **100 ms < T_{POR} < 300 ms** for a lower power-ramp rate.

Configuration Pins Power Supply

Table 9–4 lists the configuration pins for Arria II devices.

Table 9–4. Configuration pins for Arria II Devices

Devices	Configuration Pins
Arria II GX	All dedicated configuration pins are supplied by V_{CCIO} for I/O banks 3C and 8C in which they reside. The supported configuration voltages are 1.8, 2.5, 3.0, and 3.3 V. Use the V_{CCIO} pin for I/O banks 3C and 8C to power all the dedicated configuration inputs, dedicated configuration outputs, and dedicated configuration bidirectional pins that you used for configuration. With V_{CCIO} for I/O banks 3C and 8C, the configuration input buffers do not have to the share power lines with the regular I/O buffer. You must power the dual function configuration pins that you used for configuration with the V_{CCIO} power supply in which the configuration pins reside. For more information about the configuration voltage standard applied to the V_{CCIO} power supply, refer to Table 9–6 on page 9–9.
Arria II GZ	All dedicated configuration pins and dual-function pins are supplied by V_{CCPGM}. The supported configuration voltages are 1.8, 2.5, and 3.0 V. Use the V_{CCPGM} pin to power all the dedicated configuration inputs, dedicated configuration outputs, and dedicated configuration bidirectional pins that you used for configuration. With V_{CCPGM}, the configuration input buffers do not have to share the power lines with the regular I/O buffer.

Arria II devices do not support a 1.5-V configuration. The operating voltage for the configuration input pin is independent of the I/O banks power supply V_{CCIO} during configuration. Therefore, for Arria II devices, you do not require configuration voltage constraints on V_{CCIO}.

For more information, refer to the *Power Management in Arria II Devices* chapter.

For more information about the configuration pins connection recommendations, refer to the *Arria II Device Family Pin Connection Guidelines*.

V_{CCPD} Pins

Arria II devices have a dedicated programming power supply, the V_{CCPD} pins. Table 9–5 lists the power supply for Arria II devices.

Table 9–5. Power Supply for Arria II Devices

Devices	Programming Power Supply
Arria II GX	V_{CCPD} must be connected to 3.3, 3.0, or 2.5 V to power the I/O pre-drivers, HSTL/SSTL input buffers, and MSEL[3..0] pins. V_{CCPD} and V_{CCIO} for I/O banks 3C and 8C must ramp up from 0 V to the desired voltage level within 100 ms when PORSEL is low or 4 ms when PORSEL is high. If these supplies are not ramped up in this specified time, your Arria II GX device will not configure successfully. If the system cannot ramp up the power supplies within 100 ms or 4 ms, you must hold nCONFIG low until all the power supplies are stable. You must connect V_{CCPD} according to the I/O standard used in the same bank: ■ For 3.3-V I/O standards, connect V_{CCPD} to 3.3 V. ■ For 3.0-V I/O standards, connect V_{CCPD} to 3.0 V. ■ For 2.5-V and below I/O standards, connect V_{CCPD} to 2.5 V.
Arria II GZ	V_{CCPD} must be connected to 3.0 or 2.5 V to power the I/O pre-drivers and JTAG I/O pins (TCK, TMS, TDI, TDO, and TRST). V_{CCPD} and V_{CCPGM} must ramp up from 0 V to the desired voltage level within 100 ms when PORSEL is low or 4 ms when PORSEL is high. If these supplies are not ramped up in this specified time, your Arria II GZ device will not configure successfully. If the system cannot ramp up the power supplies within 100 ms or 4 ms, you must hold nCONFIG low until all the power supplies are stable. V_{CCPD} must be greater than or equal to V_{CCIO} of the same bank: ■ If the V_{CCIO} of the bank is powered to 3.0 V, V_{CCPD} must be powered up to 3.0 V. ■ If the V_{CCIO} of the bank is powered to 2.5 V or lower, V_{CCPD} must be powered up to 2.5 V.

For more information about configuration pins power supply, refer to "Device Configuration Pins" on page 9–40.

Configuration Process

The following sections describe the general configuration process for FPP, standard AS, fast AS, and PS schemes.

Power Up

To begin the configuration process, you must fully power the relevant voltage supply to the appropriate voltage levels.

For an FPP configuration in Arria II GX devices, the DATA[7..1] pins are supplied by V_{CCIO} for I/O bank 6A. You must power up this bank when you use the FPP configuration. For Arria II GZ devices, the DATA[7..1] pins are powered up by V_{CCPGM} during configuration or by V_{CCIO} if they are used as regular I/Os in user mode.

Reset

After power up, the Arria II device goes through a POR. The POR delay depends on the MSEL pin settings. During POR, the device resets, holds nSTATUS low, clears the configuration RAM bits, and tri-states all user I/O pins. After the device successfully exits POR, all user I/O pins continue to be tri-stated. While nCONFIG is low, the device is in reset. When the device comes out of reset, nCONFIG must be at a logic-high level in order for the device to release the open-drain nSTATUS pin. After nSTATUS is released, it is pulled high by a pull-up resistor and the device is ready to receive configuration data.

Before and during configuration, all user I/O pins are tri-stated. If nIO_pullup is driven low during power up and configuration, the user I/O pins and dual-purpose I/O pins have weak pull-up resistors, which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

Configuration

nCONFIG and nSTATUS must be at a logic-high level in order for the configuration stage to begin. The device receives configuration data on its DATA pins and (for synchronous configuration schemes) the clock source on the DCLK pin. Configuration data is latched into the FPGA on the rising edge of DCLK. After the FPGA has received all the configuration data successfully, it releases the CONF_DONE pin, which is pulled high by a pull-up resistor. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin.

To ensure DCLK and DATA0 are not left floating at the end of configuration, they must be driven either high or low, whichever is convenient on your board. Use the dedicated DATA[0] pin for both PS and AS configuration modes. It is not available as a user I/O pin after configuration.

For FPP and PS configuration schemes, the configuration clock (DCLK) speed must be below the specified frequency to ensure correct configuration. No maximum DCLK period exists, which means you can pause the configuration by halting DCLK for an indefinite amount of time.

A reconfiguration is initiated by toggling the nCONFIG pin from high to low and then back to high with a minimum t_{CFG} low-pulse width either in the configuration, configuration error, initialization, or user mode stage. When nCONFIG is pulled low, nSTATUS and CONF_DONE are also pulled low and all the I/O pins are tri-stated. After nCONFIG and nSTATUS return to a logic-high level, configuration begins.

Configuration Error

If an error occurs during configuration, Arria II devices assert the nSTATUS signal low, indicating a data frame error; the CONF_DONE signal stays low. If you turn on the **Auto-restart configuration after error** option (available in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box), the Arria II device resets the configuration device and retries the configuration. If you turn off this option, the system must monitor nSTATUS for errors and then pulse nCONFIG low to restart the configuration.

Initialization

In Arria II devices, the initialization clock source is either the internal oscillator or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If you use the internal oscillator, the Arria II device provides itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device. Driving DCLK to the device after configuration is complete does not affect device operation.

You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the **CLKUSR** option. You can turn on the **Enable user-supplied start-up clock (CLKUSR)** option in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box. If you supply a clock on CLKUSR, it does not affect the configuration process. After all the configuration data is accepted and CONF_DONE goes high, CLKUSR is enabled after the time specified as t_{CD2CU}. After this time period elapses, Arria II devices require a minimum number of clock cycles to initialize properly and enter user mode as specified in the t_{CD2UMC} parameter.

Two DCLK falling edges are required after CONF_DONE goes high to begin the initialization of the device for both uncompressed and compressed bitstream in the FPP or PS configuration mode.

User Mode

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box. If you use the INIT_DONE pin, it is high due to an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. After the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. When initialization is complete, the device enters user mode. In user-mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

Configuration Schemes

The following sections describe configuration schemes for Arria II devices.

MSEL Pin Settings

Select the configuration scheme by driving the Arria II device MSEL pins either high or low, as listed in Table 9–6 and Table 9–7. The MSEL input buffers are powered by the V_{CCPD} and V_{CCPGM} power supplies for Arria II GX and GZ devices, respectively. Altera recommends hardwiring the MSEL[] pins to V_{CCPD}/V_{CCPGM} or GND. The MSEL[3..0] pins have 5-kΩ internal pull-down resistors that are always active. During POR and during reconfiguration, the MSEL pins must be at LVTTL V_{IL} and V_{IH} levels to be considered logic low and logic high, respectively.

To avoid problems with detecting an incorrect configuration scheme, hardwire the MSEL[] pins to V_{CCPD}/V_{CCPGM} or GND without pull-up or pull-down resistors. Do not drive the MSEL[] pins by a microprocessor or another device.

For Figure 9–1 on page 9–12 through Figure 9–30 on page 9–67, MSEL[n..0] represents MSEL[3..0] for Arria II GX devices and MSEL[2..0] for Arria II GZ devices as listed in Table 9–6 and Table 9–7, respectively.

Table 9–6. Configuration Schemes for Arria II GX Devices (Part 1 of 2)

Configuration Scheme	MSEL3	MSEL2	MSEL1	MSEL0	POR Delay	Configuration Voltage Standard (V) *(1)*
FPP	0	0	0	0	Fast	3.3, 3.0, 2.5
	0	1	1	1	Fast	1.8
FPP with design security feature, decompression, or both enabled *(2)*	0	0	0	1	Fast	3.3, 3.0, 2.5
	1	0	0	0	Fast	1.8
PS	0	0	1	0	Fast	3.3, 3.0, 2.5
	1	0	0	1	Fast	1.8
	1	0	1	0	Standard	3.3, 3.0, 2.5
	1	0	1	1	Standard	1.8

Table 9–6. Configuration Schemes for Arria II GX Devices (Part 2 of 2)

Configuration Scheme	MSEL3	MSEL2	MSEL1	MSEL0	POR Delay	Configuration Voltage Standard (V) *(1)*
AS with or without remote system upgrade	0	0	1	1	Fast	3.3
	1	1	0	1	Fast	3.0, 2.5
	1	1	1	0	Standard	3.3
	1	1	1	1	Standard	3.0, 2.5
JTAG-based configuration *(3)*	*(4)*	*(4)*	*(4)*	*(4)*	—	—

Notes to Table 9–6:

(1) Configuration voltage standard applied to the V_{CCIO} power supply in which the configuration pins reside.

(2) These modes are only supported when using a MAX II device or a microprocessor with flash memory for configuration. In these modes, the host system must output a DCLK that is x4 the data rate.

(3) JTAG-based configuration takes precedence over other configuration schemes, which means the MSEL pin settings are ignored. JTAG-based configuration does not support the design security or decompression features.

(4) Do not leave the MSEL pins floating. Connect them to V_{CCPD} or GND. These pins support the non-JTAG configuration scheme used in production. If you only use the JTAG configuration, Altera recommends connecting the MSEL pins to GND.

Table 9–7 lists the configuration schemes for Arria II GZ devices.

Table 9–7. Configuration Schemes for Arria II GZ Devices

Configuration Scheme	MSEL2	MSEL1	MSEL0	POR Delay	Configuration Voltage Standard (V)
FPP	0	0	0	Fast/Standard	3.0, 2.5, 1.8
PS	0	1	0	Fast/Standard	3.0, 2.5, 1.8
Fast AS (40 MHz) *(1)*	0	1	1	Fast/Standard	3.0, 2.5, 1.8
Remote system upgrade fast AS (40 MHz) *(1)*	0	1	1	Fast/Standard	3.0, 2.5, 1.8
FPP with design security feature and/or decompression enabled *(2)*	0	0	1	Fast/Standard	3.0, 2.5, 1.8
JTAG-based configuration *(3)*	*(4)*	*(4)*	*(4)*	—	—

Notes to Table 9–7:

(1) Arria II GZ devices only support fast AS configuration. You must use either EPCS64 or EPCS128 devices to configure an Arria II GZ device in fast AS mode.

(2) These modes are only supported when using a MAX II device or microprocessor with flash memory for configuration. In these modes, the host system must output a DCLK that is x4 the data rate.

(3) The JTAG-based configuration takes precedence over other configuration schemes, which means the MSEL pin settings are ignored. The JTAG-based configuration does not support the design security or decompression features.

(4) Do not leave the MSEL pins floating, connect them to V_{CCPGM} or GND. These pins support non-JTAG configuration scheme used in production. If you only use the JTAG configuration, Altera recommends connecting the MSEL pins to GND.

Raw Binary File Size

Table 9–8 lists the uncompressed raw binary file (**.rbf**) configuration file sizes for Arria II devices.

Table 9–8. Uncompressed .rbf Sizes for Arria II Devices

Device	Data Size (bits)
EP2AGX45	29,599,704
EP2AGX65	29,599,704
EP2AGX95	50,376,968
EP2AGX125	50,376,968
EP2AGX190	82,763,208
EP2AGX260	82,763,208
EP2AGZ225	94,557,465
EP2AGZ300	128,395,577
EP2AGZ350	128,395,577

Use the data in Table 9–8 to estimate the file size before design compilation. Different configuration file formats, such as a hexidecimal (**.hex**) or tabular text file (**.ttf**) format, have different file sizes. For the different types of configuration file and file sizes, refer to the Quartus II software. However, for a specific version of the Quartus II software, any design targeted for the same device has the same uncompressed configuration file size. If you are using compression, the file size can vary after each compilation because the compression ratio depends on your design.

For more information about setting device configuration options or creating configuration files, refer to the *Device Configuration Options* and *Configuration File Formats* chapters in volume 2 of the *Configuration Handbook*.

Fast Passive Parallel Configuration

FPP configuration in Arria II devices is designed to meet the continuously increasing demand for faster configuration times. Arria II devices are designed with the capability of receiving byte-wide configuration data per clock cycle.

You can perform FPP configuration of Arria II devices using an intelligent host such as a MAX II device or microprocessor.

FPP Configuration Using a MAX II Device as an External Host

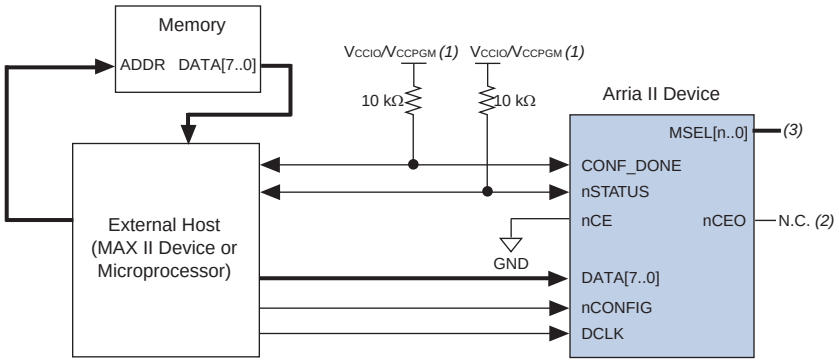
FPP configuration using compression and an external host provides the fastest method to configure Arria II devices. In this configuration scheme, you can use a MAX II device or microprocessor as an intelligent host that controls the transfer of configuration data from a storage device, such as flash memory, to the target Arria II device. You can store configuration data in **.rbf**, **.hex**, or **.ttf** format. When using the MAX II device or microprocessor as an intelligent host, a design that controls the configuration process, such as fetching the data from flash memory and sending it to the device, must be stored in the MAX II device or microprocessor.

If you use the Arria II decompression and/or design security features, the external host must send a DCLK frequency that is x4 the data rate.

The x4 DCLK signal does not require an additional pin and is sent on the DCLK pin. The maximum DCLK frequency is 125 MHz, which results in a maximum data rate of 250 Mbps. For Arria II GX devices, if you are not using the decompression or design security features, the data rate is x1 the DCLK frequency. For Arria II GZ devices, if you are not using the decompression or design security features, the data rate is x8 the DCLK frequency.

Figure 9–1 shows the configuration interface connections between an Arria II device and a MAX II device for single device configuration.

Figure 9–1. Single Device FPP Configuration Using an External Host



Notes to Figure 9–1:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for the Arria II device. For Arria II GX devices, use the V_{CCIO} pin. For Arria II GZ devices, use the V_{CCPGM} pin. V_{CCIO}/V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on both the device and the external host. Altera recommends powering up the configuration system I/Os with V_{CCIO}/V_{CCPGM}.

(2) You can leave the nCEO pin unconnected or used as a user I/O pin when it does not feed the nCE pin of the other device.

(3) The MSEL pin settings vary for different configuration voltage standards and POR delay. To connect MSEL[3..0]for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect MSEL[2..0] for an Arria II GZ device, refer to Table 9–7 on page 9–10.

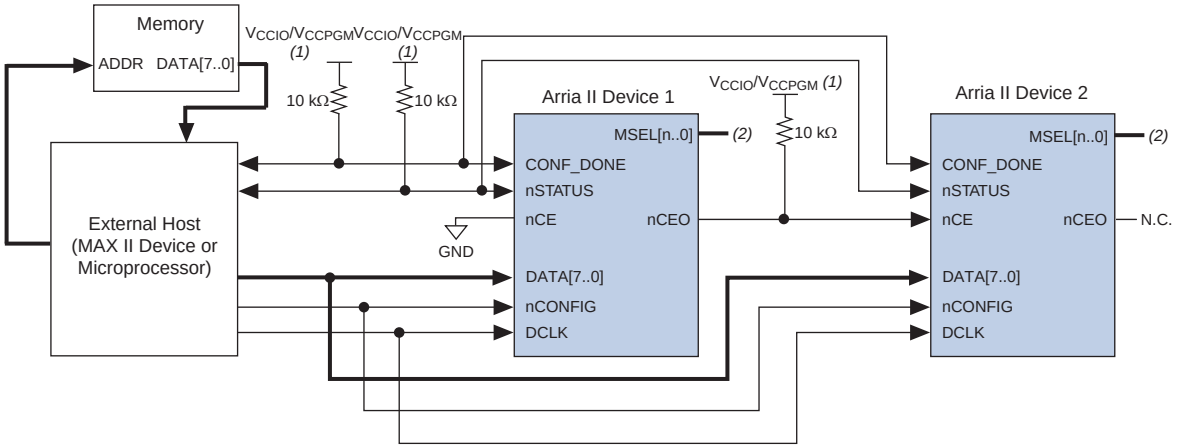
Arria II devices receive configuration data on the DATA[7..0] pins and the clock is received on the DCLK pin. Data is latched into the device on the rising edge of DCLK. If you are using the Arria II decompression and/or design security features, configuration data is latched on the rising edge of every fourth DCLK cycle. After the configuration data is latched in, it is processed during the following three DCLK cycles. Therefore, you can only stop DCLK after three clock cycles after the last data is latched into the Arria II device.

In Arria II devices, the initialization clock source is either the internal oscillator or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If you use the internal oscillator, the Arria II device provides itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device. Driving DCLK to the device after configuration is complete does not affect device operation.

You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the **CLKUSR** option. You can turn on the **Enable user-supplied start-up clock (CLKUSR)** option in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box. If you supply a clock on CLKUSR, it does not affect the configuration process. Arria II devices support an f_{MAX} of 125 MHz.

Figure 9–2 shows how to configure multiple Arria II devices using a MAX II device. This circuit is similar to the FPP configuration circuit for a single device, except the Arria II devices are cascaded for multi-device configuration.

Figure 9–2. Multi-Device FPP Configuration Using an External Host



Notes to Figure 9–2:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for the Arria II device. For Arria II GX devices, use the V_{CCIO} pin. For Arria II GZ devices, use the V_{CCPGM} pin. V_{CCIO}/V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on both the device and the external host. Altera recommends powering up the configuration system I/Os with V_{CCIO}/V_{CCPGM}.

(2) The MSEL pin settings vary for different configuration voltage standards and POR delay. To connect MSEL[3..0] for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect MSEL[2..0] for an Arria II GZ device, refer to Table 9–7 on page 9–10.

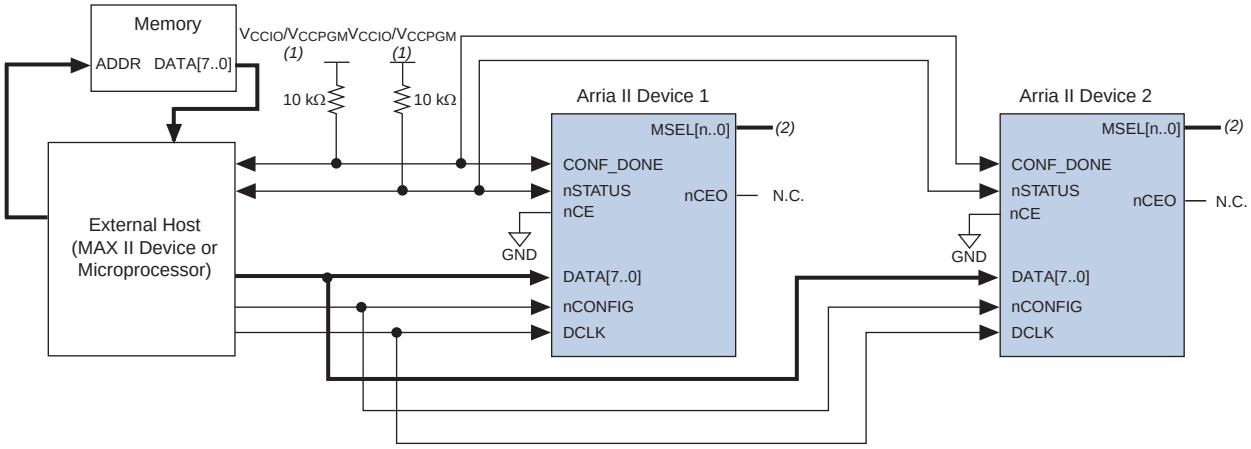
After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the nCE pin of the second device, which prompts the second device to begin configuration. The second device in the chain begins configuration in one clock cycle; therefore, the transfer of data destinations is transparent to the MAX II device or microprocessor. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. The configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

All nSTATUS and CONF_DONE pins are tied together and if any device detects an error, configuration stops for the entire chain and you must reconfigure the entire chain. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

If a system has multiple devices that contain the same configuration data, tie all device nCE inputs to GND and leave the nCEO pins floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. Configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices start and complete configuration at the same time.

Figure 9–3 shows a multi-device FPP configuration when both Arria II devices are receiving the same configuration data.

Figure 9–3. Multiple-Device FPP Configuration Using an External Host When Both Devices Receive the Same Data



Notes to Figure 9–3:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for the Arria II device. For Arria II GX devices, use the V_{CCIO} pin. For Arria II GZ devices, use the V_{CCPGM} pin. V_{CCIO}/V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on both the device and the external host. Altera recommends powering up the configuration system I/Os with V_{CCIO}/V_{CCPGM}.

(2) The MSEL pin settings vary for different configuration voltage standards and POR delay. To connect MSEL[3..0] for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect MSEL[2..0] for an Arria II GZ device, refer to Table 9–7 on page 9–10.

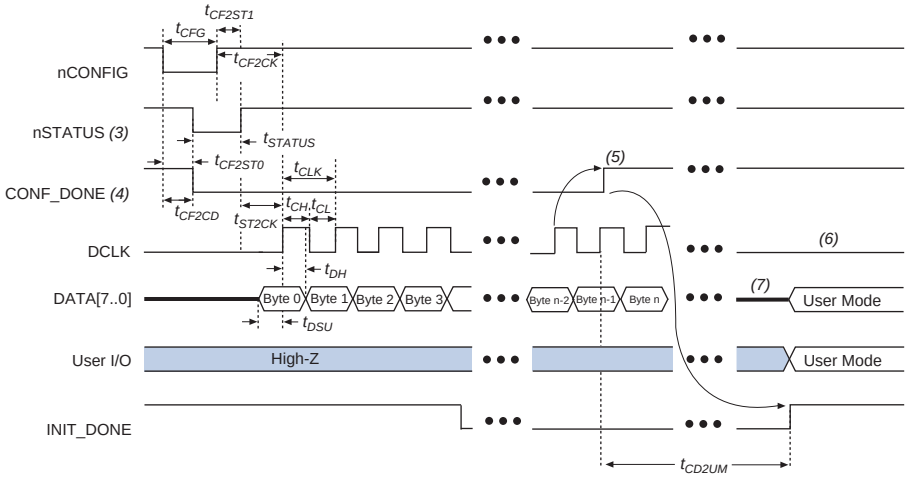
You can use a single configuration chain to configure Arria II devices with other Altera devices that support FPP configuration. To ensure that all devices in the chain complete configuration at the same time, or that an error flagged by one device initiates reconfiguration in all devices, tie all of the device CONF_DONE and nSTATUS pins together.

For more information about configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook*.

FPP Configuration Timing

Figure 9–4 shows the timing waveform for an FPP configuration when using a MAX II device as an external host. This waveform shows timing when the decompression and design security features are not enabled.

Figure 9–4. FPP Configuration Timing Waveform with Decompression and Design Security not Enabled *(Note 1), (2)*



Notes to Figure 9–4:

(1) Use this timing waveform when you do not use the decompression and design security features.

(2) The beginning of this waveform shows the device in user mode. In user mode, nCONFIG, nSTATUS, and CONF_DONE are at logic-high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.

(3) After power-up, the Arria II device holds nSTATUS low for the time of the POR delay.

(4) After power-up, before and during configuration, CONF_DONE is low.

(5) Two DCLK falling edges are required after CONF_DONE goes high to begin the initialization of the device.

(6) Do not leave DCLK floating after configuration. You can drive it high or low, whichever is more convenient.

(7) DATA[7..1] are available as user I/O pins after configuration. The state of these pins depends on the dual-purpose pin settings. For Arria II GX devices, DATA[0] is a dedicated pin that is used for both the PS and AS configuration modes and is not available as a user I/O pin after configuration. For Arria II GZ devices, DATA[0] is available as a user I/O pin after configuration.

Table 9–9 lists the timing parameters for Arria II devices for an FPP configuration when you do not enable the decompression and design security features.

Table 9–9. FPP Timing Parameters for Arria II Devices with Decompression and Design Security not Enabled *(Note 1)*

Symbol	Parameter	Minimum	Maximum	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low	—	800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low	—	800	ns
t_{CFG}	nCONFIG low pulse width	2	—	µs
t_{STATUS}	nSTATUS low pulse width	10	500 *(3)*	µs
t_{CF2ST1} *(2)*	nCONFIG high to nSTATUS high	—	500 *(3)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	500	—	µs
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2	—	µs
t_{DSU}	Data setup time before rising edge on DCLK	4	—	ns
t_{DH}	Data hold time after rising edge on DCLK	0 *(4)*	—	ns
t_{CH}	DCLK high time	3.2 *(4)*	—	ns
t_{CL}	DCLK low time	3.2 *(4)*	—	ns
t_{CLK}	DCLK period	8	—	ns
f_{MAX}	DCLK frequency	—	125	MHz
t_R	Input rise time	—	40	ns
t	Input fall time	—	40	ns
t_{CD2UM}	CONF_DONE high to user mode *(5)*	55	150	ns
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period	—	—
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	t_{CD2CU} + (8532 × CLKUSR period)	—	—

Notes to Table 9–9:

(1) Use these timing parameters when you do not enable the decompression and design security features.

(2) This value is applicable if you do not delay configuration by externally holding the nSTATUS low.

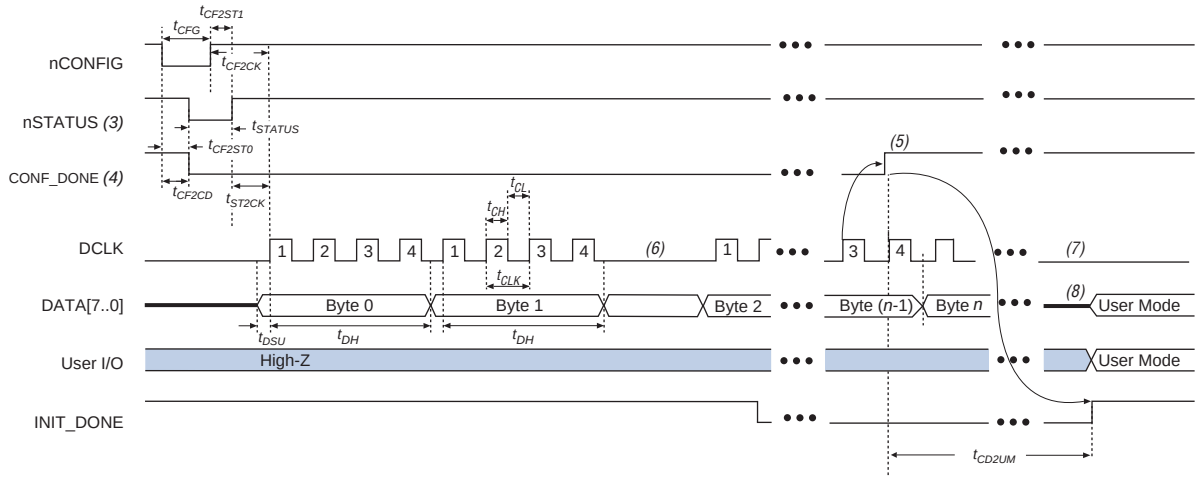
(3) You can obtain this value if you do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(4) The values listed for t_{DH}, t_{CH}, and t_{CL} are applicable only for Arria II GX devices. For Arria II GZ devices, t_{DH} = 1 ns, t_{CH} = 3.6 ns, and t_{CL} = 3.6 ns, respectively.

(5) The minimum and maximum numbers apply only if you chose the internal oscillator as the clock source for initializing the device.

Figure 9–5 shows the timing waveform for an FPP configuration when using a MAX II device or microprocessor as an external host. This waveform shows timing when you enable the decompression, the design security features, or both.

Figure 9–5. FPP Configuration Timing Waveform with Decompression or Design Security Enabled *(Note 1), (2)*



Notes to Figure 9–5:

(1) Use this timing waveform when you use the decompression and/or design security features.

(2) The beginning of this waveform shows the device in user-mode. In user-mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.

(3) After power-up, the Arria II GX device holds nSTATUS low for the time of the POR delay.

(4) After power-up, before and during configuration, CONF_DONE is low.

(5) Two DCLK falling edges are required after CONF_DONE goes high to begin the initialization of the device.

(6) If required, you can pause DCLK by holding it low. When DCLK restarts, the external host must provide data on the DATA[7..0] pins prior to sending the first DCLK rising edge.

(7) Do not leave DCLK floating after configuration. You can drive it high or low, whichever is more convenient.

(8) DATA[7..1] are available as user I/O pins after configuration. The state of these pins depends on the dual-purpose pin settings. For Arria II GX devices, DATA[0] is a dedicated pin that is used for both the PS and AS configuration modes and is not available as a user I/O pin after configuration. For Arria II GZ devices, DATA[0] is available as a user I/O pin after configuration.

Table 9–10 lists the timing parameters for Arria II devices for an FPP configuration when you enable the decompression, the design security features, or both.

Table 9–10. FPP Timing Parameters for Arria II GX Devices with the Decompression or Design Security Features Enabled *(Note 1)*

Symbol	Parameter	Minimum	Maximum	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low	—	800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low	—	800	ns
t_{CFG}	nCONFIG low pulse width	2	—	µs
t_{STATUS}	nSTATUS low pulse width	10	500 *(3)*	µs
t_{CF2ST1} *(2)*	nCONFIG high to nSTATUS high	—	500 *(3)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	500	—	µs
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2	—	µs
t_{DSU}	Data setup time before rising edge on DCLK	4	—	ns
t_{DH}	Data hold time after rising edge on DCLK	24 *(4)*	—	ns
t_{CH}	DCLK high time	3.2 *(4)*	—	ns
t_{CL}	DCLK low time	3.2 *(4)*	—	ns
t_{CLK}	DCLK period	8	—	ns
f_{MAX}	DCLK frequency	—	125	MHz
t_{DATA}	Data rate	—	250	Mbps
t_R	Input rise time	—	40	ns
t	Input fall time	—	40	ns
t_{CD2UM}	CONF_DONE high to user mode *(5)*	55	150	ns
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period	—	—
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	t_{CD2CU} + (8532 × CLKUSR period)	—	—

Notes to Table 9–10:

(1) Use these timing parameters when you enable the decompression and design security features.

(2) This value is applicable if you do not delay configuration by externally holding the nSTATUS low.

(3) You can obtain this value if you do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(4) The values listed for t_{DH}, t_{CH}, and t_{CL} are applicable only for Arria II GX devices. For Arria II GZ devices, t_{DH} = 3/(DCLK frequency) + 1, t_{CH} = 3.6 ns, and t_{CL} = 3.6 ns, respectively.

(5) The minimum and maximum numbers apply only if you choose the internal oscillator as the clock source for initializing the device.

For more information about setting device configuration options or creating configuration files, refer to the *Device Configuration Options* and *Configuration File Formats* chapters in volume 2 of the *Configuration Handbook*.

AS and Fast AS Configuration (Serial Configuration Devices)

Arria II GX and GZ devices are configured using a serial configuration device in the AS configuration scheme and the fast AS configuration scheme, respectively. These configuration devices are low-cost devices with non-volatile memory that feature a simple four-pin interface and a small form factor. These features make serial configuration devices an ideal low-cost configuration solution.

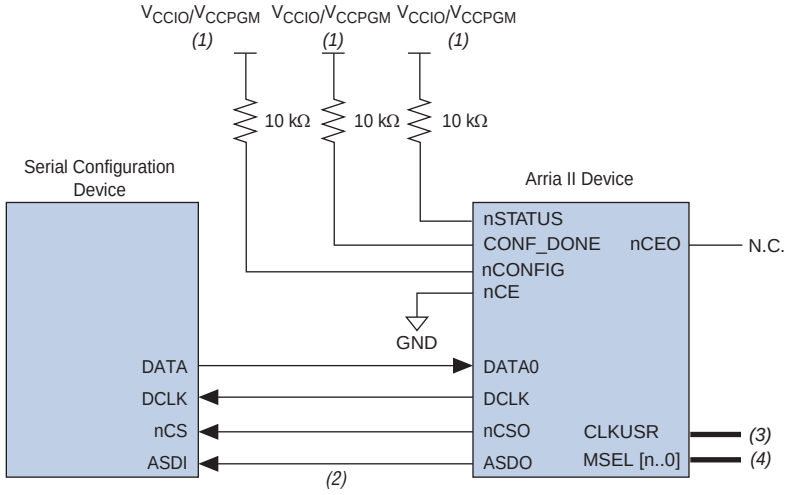
For more information about serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

Serial configuration devices provide a serial interface to access configuration data. During device configuration, Arria II devices read configuration data using the serial interface, decompress data if necessary, and configure their SRAM cells. This scheme is referred to as the AS configuration scheme because the Arria II device controls the configuration interface. This scheme contrasts with the PS configuration scheme, where the configuration device controls the interface.

The Arria II decompression and design security features are available when configuring your Arria II GX device using AS mode and when configuring your Arria II GZ device using fast AS mode.

Serial configuration devices have a four-pin interface—serial clock input (DCLK), serial data output (DATA), AS data input (ASDI), and an active-low chip select (nCS). This four-pin interface connects to the Arria II device pins, as shown in Figure 9–6.

Figure 9–6. Single Device AS Configuration



Notes to Figure 9–6:

(1) Connect the pull-up resistors to the V_{CCIO} power supply of bank 3C for Arria II GX devices and to V_{CCPGM} at a 3.0-V power supply for Arria II GZ devices.

(2) Arria II devices use the ASDO-to-ASDI path to control the configuration device.

(3) Arria II devices have an option to select **CLKUSR** (40 MHz maximum) as the external clock source for DCLK.

(4) The MSEL pin settings vary for different configuration voltage standards and POR delay. To connect MSEL[3..0]for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect MSEL[2..0] for an Arria II GZ device, refer to Table 9–7 on page 9–10.

The serial clock (DCLK) generated by the Arria II device controls the entire configuration cycle and provides timing for the serial interface. During the configuration, Arria II devices use an internal oscillator or an external clock source to generate DCLK. At the initial stage of the configuration cycle, the Arria II device generates a default DCLK (40 MHz maximum) from the internal oscillator to read the header information of the programming data stored in the EPCS. After the header information is read from the EPCS, depending on the clock source being selected, the configuration cycle continues with a slow clock (20 MHz maximum) or a fast clock (40 MHz maximum) from the internal oscillator or an external clock from **CLKUSR** (40 MHz maximum). You can change the clock source option in the Quartus II software from the **Configuration** tab of the **Device and Pin Options** dialog box.

☞ Arria II GZ devices only support fast AS configuration (40 MHz maximum) and do not support a slow clock.

In AS and fast AS configuration schemes, Arria II devices drive out control signals on the falling edge of DCLK. The serial configuration device responds to the instructions by driving out configuration data on the falling edge of DCLK. Then the data is latched into the Arria II device on the following falling edge of DCLK.

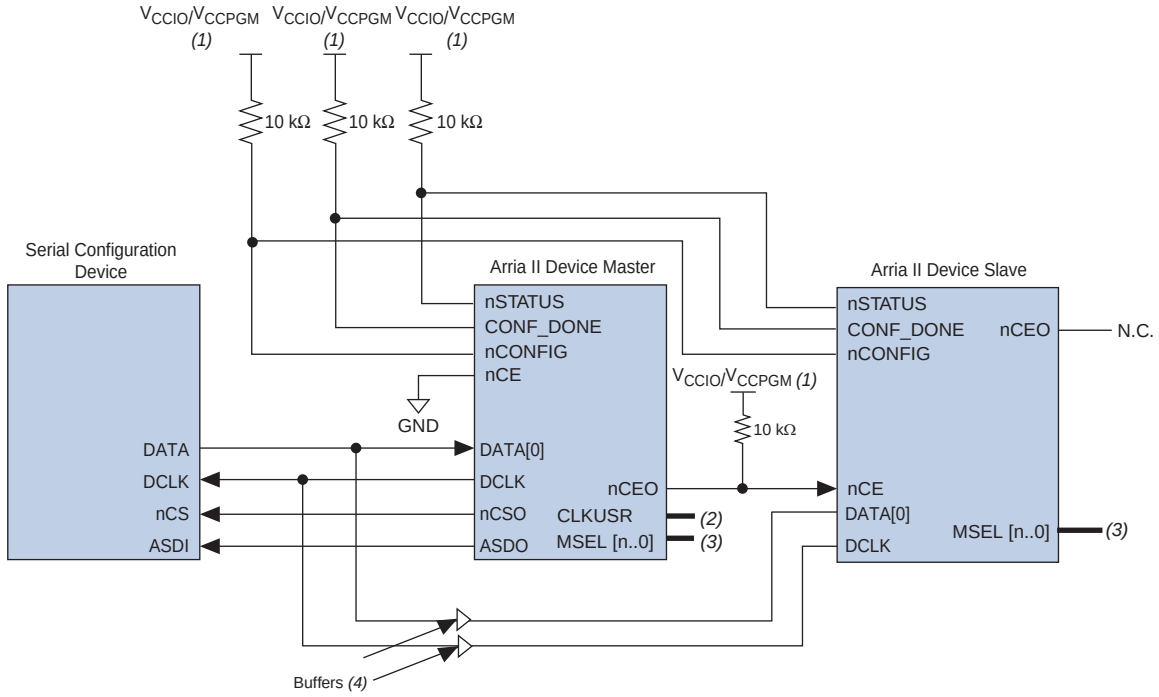
In configuration mode, Arria II devices enable the serial configuration device by driving the nCSO output pin low, which connects to the chip select (nCS) pin of the configuration device. The Arria II device uses the serial clock (DCLK) and serial data output (ASDO) pins to send operation commands, read address signals, or both, to the serial configuration device. The configuration device provides data on its serial data output (DATA) pin, which connects to the DATA0 input of the Arria II devices.

You can configure multiple Arria II devices using a single serial configuration device. Cascade multiple Arria II devices using the chip-enable (nCE) and chip-enable-out (nCEO) pins. The first device in the chain must have its nCE pin connected to GND. You must connect its nCEO pin to the nCE pin of the next device in the chain. When the first device captures all its configuration data from the bitstream, it drives the nCEO pin low, enabling the next device in the chain. You must leave the nCEO pin of the last device unconnected. The nCONFIG, nSTATUS, CONF_DONE, DCLK, and DATA0 pins of each device in the chain are connected (refer to Figure 9–7).

The first Arria II device in the chain is the configuration master and controls configuration of the entire chain. You must connect its MSEL pins to select the AS configuration scheme. The remaining Arria II devices are configuration slaves. You must connect their MSEL pins to select the PS configuration scheme. Any other Altera device that supports PS configuration can also be part of the chain as a configuration slave.

Figure 9–7 shows the pin connections for the multi-device AS configuration.

Figure 9–7. Multi-Device AS Configuration



Notes to Figure 9–7:

(1) Connect the pull-up resistors to the V_{CCIO} power supply of the I/O bank 3C for Arria II GX devices and to V_{CCPGM} at a 3.0-V power supply for Arria II GZ devices.

(2) Arria II devices have an option to select **CLKUSR** (40 MHz maximum) as the external clock source for DCLK.

(3) The MSEL pin settings vary for different configuration voltage standards and POR delay. To connect MSEL[3..0]for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect MSEL[2..0] for an Arria II GZ device, refer to Table 9–7 on page 9–10.

(4) Connect the repeater buffers between the Arria II master and slave devices for DATA[0] and DCLK. This is to prevent any potential signal integrity and clock skew problems.

The timing parameters for AS mode are not listed here because the t_{CF2CD}, t_{CF2ST0}, t_{CFG}, t_{STATUS}, t_{CF2ST1}, and t_{CD2UM} timing parameters are identical to the timing parameters for PS mode listed in Table 9–12 on page 9–30.

As shown in Figure 9–7, the nSTATUS and CONF_DONE pins on all target devices are connected together with external pull-up resistors. These pins are open-drain bidirectional pins on the devices. When the first device asserts nCEO (after receiving all its configuration data), it releases its CONF_DONE pin. But the subsequent devices in the chain keep this shared CONF_DONE line low until they have received their configuration data. When all target devices in the chain have received their configuration data and have released CONF_DONE, the pull-up resistor drives a high level on this line and all devices simultaneously enter initialization mode.

While you can cascade Arria II devices, you cannot cascade or chain together serial configuration devices.

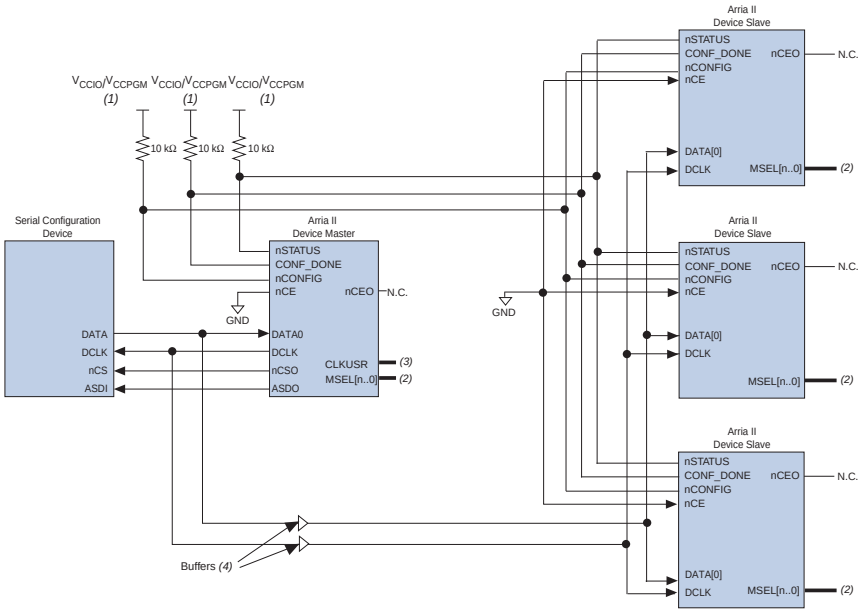
If the configuration bitstream size exceeds the capacity of a serial configuration device, you must select a larger configuration device, enable the compression feature, or both. When configuring multiple devices, the size of the bitstream is the sum of the configuration bitstreams of the individual devices.

A system may have multiple devices that contain the same configuration data. In AS chains, you can implement this by storing one copy of the SRAM object file (**.sof**) in the serial configuration device. The same copy of the **.sof** configures the master Arria II device and all remaining slave devices concurrently. All Arria II devices must be the same density and package.

To configure four identical Arria II devices with the same **.sof**, you can set up the chain similar to the example shown in Figure 9–8. The first device is the master device and its MSEL pins must be set to select AS configuration. The other three slave devices are set up for concurrent configuration and their MSEL pins must be set to select PS configuration. The nCE input pins from the master and slave are connected to GND, and the DATA and DCLK pins connect in parallel to all four devices. During the configuration cycle, the master device reads its configuration data from the serial configuration device and transmits the configuration data to all three slave devices, configuring all of them simultaneously.

Figure 9–8 shows the multi-device AS configuration when the devices receive the same data using a single **.sof**.

Figure 9–8. Multi-Device AS Configuration When the Devices Receive the Same Data Using a Single .sof



Notes to Figure 9–8:

(1) Connect the pull-up resistors to the V_{CCIO} power supply of I/O bank 3C for Arria II GX devices and to V_{CCPGM} at a 3.0-V power supply for Arria II GZ devices.

(2) The MSEL pin settings vary for different configuration voltage standards and POR delay. To connect MSEL[3..0] for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect MSEL[2..0] for an Arria II GZ device, refer to Table 9–7 on page 9–10.

(3) Arria II devices have an option to select **CLKUSR** (40 MHz maximum) as the external clock source for DCLK.

(4) Connect the repeater buffers between the Arria II master and slave devices for DATA[0] and DCLK. This is to prevent any potential signal integrity and clock skew problems.

Guidelines for Connecting Serial Configuration Device to Arria II Devices on an AS Interface

For single- and multi-device AS configurations, the board trace length and loading between the supported serial configuration device and the Arria II devices must follow the recommendations listed in Table 9–11.

Table 9–11. Maximum Trace Length and Loading for the AS Configuration in Arria II Devices

Arria II Device AS Pins	Maximum Board Trace Length from the Arria II Device to the Serial Configuration Device (Inches)	Maximum Board Load (pF)
DCLK	10	15
DATA[0]	10	30
nCSO	10	30
ASDO	10	30

Estimating the AS Configuration Time

AS configuration time is dominated by the time it takes to transfer data from the serial configuration device to the Arria II device. This serial interface is clocked by the Arria II DCLK output (generated from an internal oscillator or an option to select **CLKUSR** as external clock source). Arria II devices support DCLK up to 40 MHz (25 ns).

Therefore, you can estimate the minimum configuration time as the following:

RBF Size × (minimum DCLK period / 1 bit per DCLK cycle) = estimated minimum configuration time.

Enabling compression reduces the amount of configuration data that is transmitted to the Arria II device, which also reduces configuration time. On average, compression reduces configuration time, depending on your design.

Programming Serial Configuration Devices

Serial configuration devices are non-volatile, flash-memory-based devices. You can program these devices in-system using an USB-Blaster™, EthernetBlaster, EthernetBlaster II, or ByteBlaster™ II download cables. Alternatively, you can program them using a microprocessor with the SRunner software driver.

To gain control of the serial configuration device pins, hold the nCONFIG pin low and pull the nCE pin high. This causes the device to reset and tri-state the AS configuration pins.

You can perform in-system programming of serial configuration devices using the conventional AS programming interface or JTAG interface solution.

Because serial configuration devices do not support the JTAG interface, the conventional method to program them is using the AS programming interface. The configuration data used to program serial configuration devices is downloaded using programming hardware.

During in-system programming, the download cable disables device access to the AS interface by driving the nCE pin high. Arria II devices are also held in reset mode by a low level on nCONFIG. After programming is complete, the download cable releases nCE and nCONFIG, allowing the pull-down and pull-up resistors to drive GND and logic high.

Altera has developed the serial flash loader (SFL); an in-system programming solution for serial configuration devices using the JTAG interface. This solution requires the Arria II device to be a bridge between the JTAG interface and the serial configuration device.

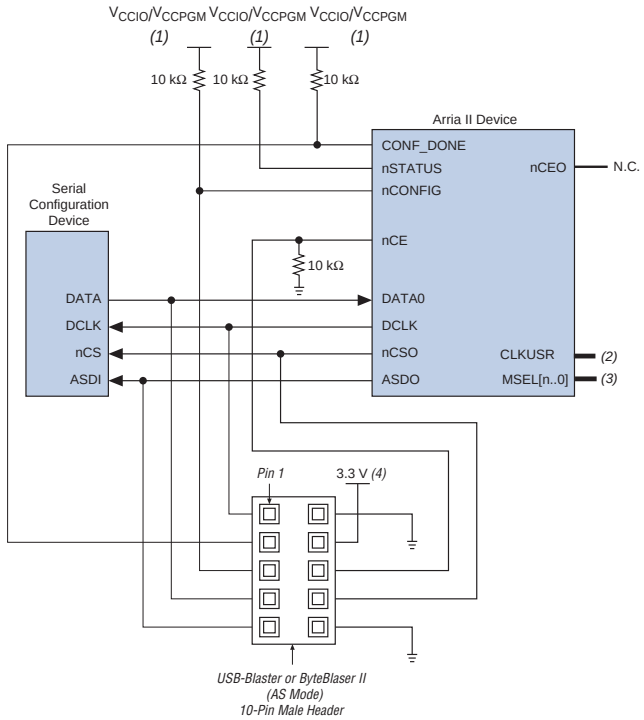
For more information about SFL, refer to *AN 370: Using the Serial FlashLoader with Quartus II Software*.

For more information, refer to the following:

- *ByteBlaster II Download Cable User Guide*
- *EthernetBlaster Communications Cable User Guide*
- *EthernetBlaster II Communications Cable User Guide*
- *USB-Blaster Download Cable User Guide*

Figure 9–9 shows the download cable connections to the serial configuration device.

Figure 9–9. In-System Programming of Serial Configuration Devices



Notes to Figure 9–9:

(1) Connect the pull-up resistors to the V_{CCIO} power supply of the I/O bank 3C for Arria II GX devices and to V_{CCPGM} at a 3.0-V power supply for Arria II GZ devices.

(2) Arria II devices have an option to select **CLKUSR** (40 MHz maximum) as the external clock source for DCLK.

(3) The MSEL pin settings vary for different configuration voltage standards and POR delay. To connect MSEL[3..0]for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect MSEL[2..0] for an Arria II GZ device, refer to Table 9–7 on page 9–10.

(4) Power up the USB-ByteBlaster, ByteBlaster II, EthernetBlaster, or EthernetBlaster II cable's $V_{CC(TRGT)}$ with V_{CCIO} 3.3 V for Arria II GX device and V_{CCPGM} 3.0 V for Arria II GZ device.

You can program serial configuration devices with the Quartus II software using the Altera programming hardware and the appropriate configuration device programming adapter.

In production environments, you can program serial configuration devices using multiple methods. You can use Altera programming hardware or other third-party programming hardware to program blank serial configuration devices before they are mounted onto PCBs. Alternatively, you can use an on-board microprocessor to program the serial configuration device in-system using C-based SRunner software drivers provided by Altera.

You can program a serial configuration device in-system by an external microprocessor using SRunner. SRunner is a software driver developed for embedded serial configuration device programming, which can be easily customized to fit in different embedded systems. SRunner is able to read a raw programming data file (.**rpd**) and write to serial configuration devices. The serial configuration device programming time using SRunner is comparable to the programming time with the Quartus II software.

For more information about SRunner, refer to *AN 418: SRunner: An Embedded Solution for EPCS Programming* and the source code on the Altera website.

For more information about programming serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

PS Configuration

You can program a PS configuration of Arria II devices using an intelligent host, such as a MAX II device or microprocessor with flash memory, or a download cable. In the PS scheme, an external host (a MAX II device, embedded processor, or host PC) controls configuration. Configuration data is clocked into the target Arria II device using the DATA0 pin at each rising edge of DCLK.

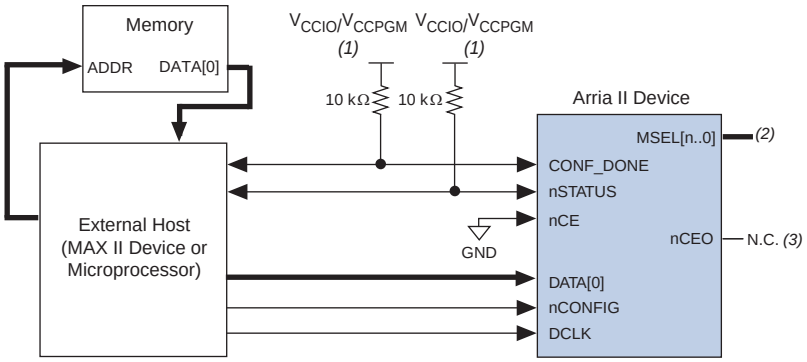
The Arria II decompression and design security features are available when configuring your Arria II device using PS mode.

PS Configuration Using a MAX II Device as an External Host

In this configuration scheme, you can use a MAX II device as an intelligent host that controls the transfer of configuration data from a storage device, such as flash memory, to the target Arria II device. You can store configuration data in **.rbf**, **.hex**, or **.ttf** format.

Figure 9–10 shows the configuration interface connections between an Arria II device and a MAX II device for single device configuration.

Figure 9–10. Single Device PS Configuration Using an External Host



Notes to Figure 9–10:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for the Arria II device. For Arria II GX devices, use the V_{CCIO} pin. For Arria II GZ devices, use the V_{CCPGM} pin. V_{CCIO}/V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on both the device and the external host. Altera recommends powering the configuration system I/Os with V_{CCIO}/V_{CCPGM}.

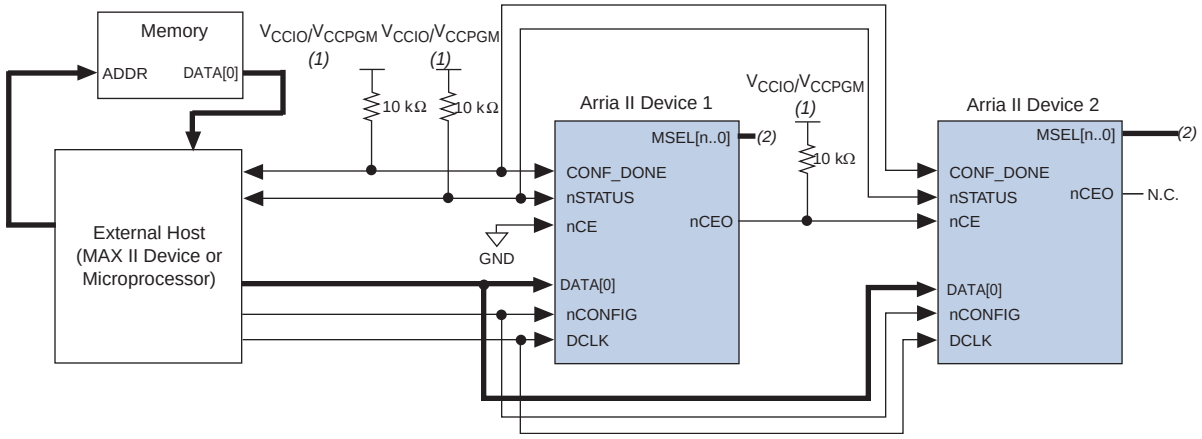
(2) The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect MSEL[3..0]for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect MSEL[2..0] for an Arria II GZ device, refer to Table 9–7 on page 9–10.

(3) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed the nCE pin of the other device.

The Arria II device receives configuration data on the DATA0 pin and the clock is received on the DCLK pin. Data is latched into the device on the rising edge of DCLK. If you are using configuration data in **.rbf**, **.hex**, or **.ttf** format, you must send the LSB of each data byte first. For example, if the **.rbf** contains the byte sequence 02 1B EE 01 FA, the serial bitstream you must transmit to the device is 0100-0000 1101-1000 0111-0111 1000-0000 0101-1111.

Figure 9–11 shows how to configure multiple devices using an external host. This circuit is similar to the PS configuration circuit for a single device, except the Arria II devices are cascaded for multi-device configuration.

Figure 9–11. Multi-Device PS Configuration Using an External Host



Notes to Figure 9–11:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for the Arria II device. For Arria II GX devices, use the V_{CCIO} pin. For Arria II GZ devices, use the V_{CCPGM} pin. V_{CCIO}/V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on both the device and the external host. Altera recommends powering up the configuration system I/Os with V_{CCIO}/V_{CCPGM}.

(2) The MSEL pin settings vary for different configuration voltage standards and POR delay. To connect MSEL[3..0] for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect MSEL[2..0] for an Arria II GZ device, refer to Table 9–7 on page 9–10.

In Arria II devices, the initialization clock source is either the internal oscillator or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If you use the internal oscillator, the Arria II device provides itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device. Driving DCLK to the device after configuration is complete does not affect device operation.

You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the **CLKUSR** option. You can turn on the **Enable user-supplied start-up clock (CLKUSR)** option in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box. If you supply a clock on CLKUSR, it does not affect the configuration process. Arria II devices support f_{MAX} of 125 MHz.

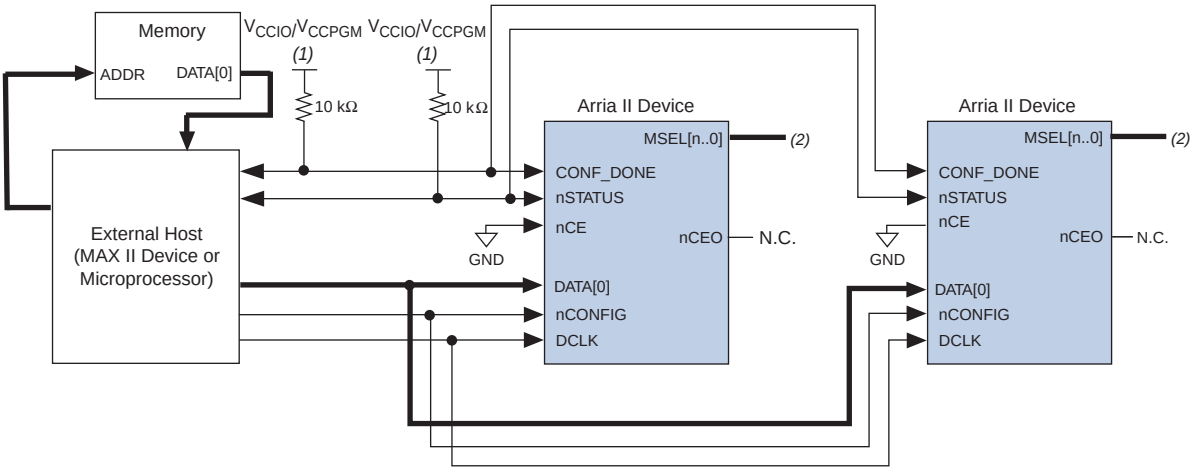
After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration in one clock cycle. Therefore, the transfer of data destinations is transparent to the MAX II device or microprocessor. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Because all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and you must reconfigure the entire chain. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

In your system, you can have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while the nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices start and complete configuration at the same time.

Figure 9–12 shows a multi-device PS configuration when both Arria II devices are receiving the same configuration data.

Figure 9–12. Multiple-Device PS Configuration When Both Devices Receive the Same Data



Notes to Figure 9–12:

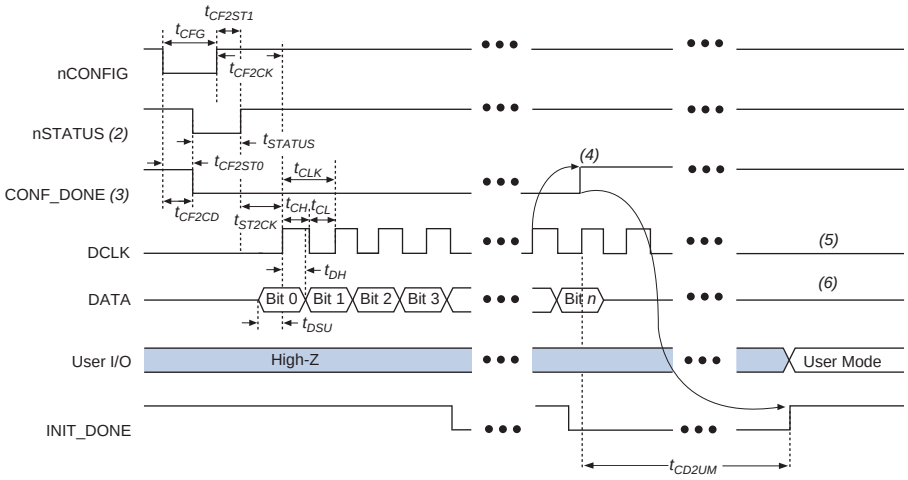
(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for the Arria II device. For Arria II GX devices, use the V_{CCIO} pin. For Arria II GZ devices, use the V_{CCPGM} pin. V_{CCIO}/V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on both the device and the external host. Altera recommends powering up the configuration system I/Os with V_{CCIO}/V_{CCPGM}.

(2) The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect MSEL[3..0]for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect MSEL[2..0] for an Arria II GZ device, refer to Table 9–7 on page 9–10.

PS Configuration Timing

Figure 9–13 shows the timing waveform for a PS configuration when using a MAX II device or microprocessor as an external host.

Figure 9–13. PS Configuration Timing Waveform ***(Note 1)***



Notes to Figure 9–13:

(1) The beginning of this waveform shows the device in user mode. In user mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.

(2) After power-up, the Arria II device holds nSTATUS low for the time of the POR delay.

(3) After power-up, before and during configuration, CONF_DONE is low.

(4) Two DCLK falling edges are required after CONF_DONE goes high to begin initialization of the device.

(5) Do not leave DCLK floating after configuration. You can drive it high or low, whichever is more convenient.

(6) For Arria II GX devices, DATA[0]is a dedicated pin that is used for both PS and AS configuration modes and is not available as a user I/O pin after configuration. For Arria II GZ devices, DATA[0] is available as a user I/O pin after configuration.

Table 9–12 lists the timing parameters for Arria II devices for PS configuration.

Table 9–12. PS Timing Parameters for Arria II Devices (Part 1 of 2)

Symbol	Parameter	Minimum	Maximum	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low	—	800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low	—	800 *(2)*	ns
t_{CFG}	nCONFIG low pulse width	2	—	µs
t_{STATUS}	nSTATUS low pulse width	10	500 *(2)*	µs
t_{CF2ST1} *(1)*	nCONFIG high to nSTATUS high	—	500 *(2)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	500	—	µs
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2	—	µs
t_{DSU}	Data setup time before rising edge on DCLK	4	—	ns
t_{DH}	Data hold time after rising edge on DCLK	0	—	ns
t_{CH}	DCLK high time	3.2	—	ns
t_{CL}	DCLK low time	3.2	—	ns
t_{CLK}	DCLK period	8	—	ns

Table 9–12. PS Timing Parameters for Arria II Devices (Part 2 of 2)

Symbol	Parameter	Minimum	Maximum	Units
f_{MAX}	DCLK frequency	—	125	MHz
t_R	Input rise time	—	40	ns
t	Input fall time	—	40	ns
t_{CD2UM}	CONF_DONE high to user mode *(3)*	55	150	ns
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period	—	—
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	t_{CD2CU} + (8532 CLKUSR period)	—	—

Notes to Table 9–12:

(1) This value is applicable if you do not delay configuration by externally holding the nSTATUS low.
(2) This value is applicable if you do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.
(3) The minimum and maximum numbers apply only if you choose the internal oscillator as the clock source for initializing the device.

For more information about device configuration options and how to create configuration files, refer to the *Device Configuration Options* and *Configuration File Formats* chapters in volume 2 of the *Configuration Handbook*.

PS Configuration Using a Download Cable

In this section, the generic term "download cable" includes the Altera USB-Blaster USB port download cable, ByteBlaster II parallel port download cable, EthernetBlaster download cable, and EthernetBlaster II download cable.

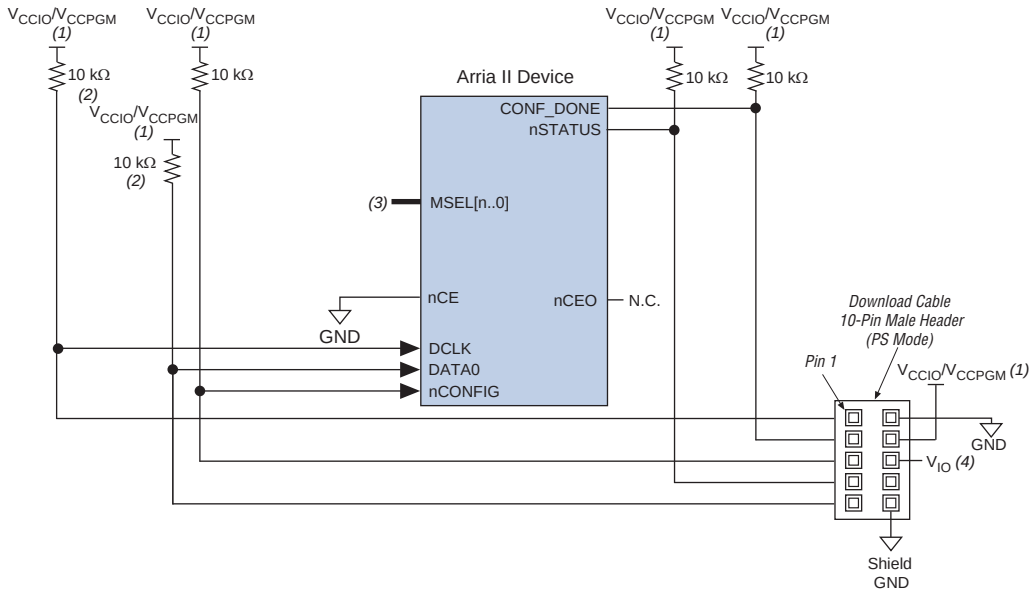
In a PS configuration with a download cable, an intelligent host (such as a PC) transfers data from a storage device to the Arria II device using the download cable.

During configuration, the programming hardware or download cable places the configuration data one bit at a time on the device's DATA0 pin. The configuration data is clocked into the target device until CONF_DONE goes high.

When using a download cable, setting the **Auto-restart configuration after error** option does not affect the configuration cycle because you must manually restart the configuration in the Quartus II software when an error occurs. Additionally, the **Enable user-supplied start-up clock** (**CLKUSR**) option has no affect on the device initialization because this option is disabled in the **.sof** when programming the device using the Quartus II programmer and download cable. Therefore, if you turn on the **CLKUSR** option, you are not required to provide a clock on the CLKUSR pin when you are configuring the device with the Quartus II programmer and a download cable.

Figure 9–14 shows a PS configuration for Arria II devices using a USB-Blaster, EthernetBlaster, EthernetBlaster II, or ByteBlaster II cable.

Figure 9–14. PS Configuration Using a USB-Blaster, EthernetBlaster, EthernetBlaster II, or ByteBlaster II Cable



Notes to Figure 9–14:

(1) Connect the pull-up resistor to the same supply voltage, V_{CCIO} for Arria II GX devices or V_{CCPGM} for Arria II GZ devices as the USB-Blaster, EthernetBlaster, EthernetBlaster II, or ByteBlaster II cable.

(2) You only need the pull-up resistors on DATA0 and DCLK if the download cable is the only configuration scheme used on your board. This ensures that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, you do not need the pull-up resistors on DATA0 and DCLK.

(3) The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect MSEL[3..0]for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect MSEL[2..0] for an Arria II GZ device, refer to Table 9–7 on page 9–10.

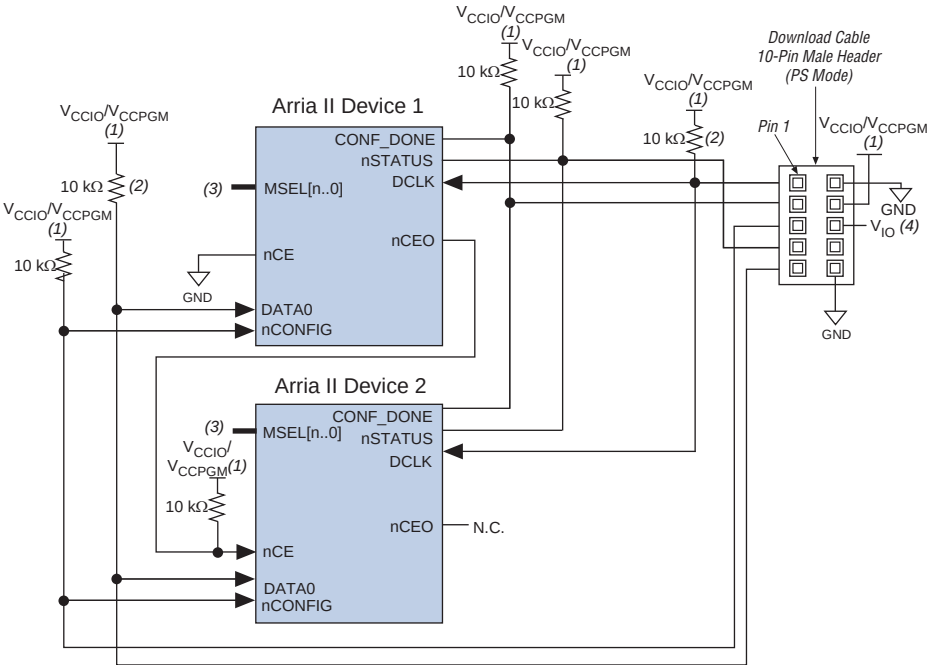
(4) In the USB-Blaster and ByteBlaster II cables, this pin is connected to nCE when it is used for AS programming; otherwise, it is a no connect.

You can use a download cable to configure multiple Arria II devices by connecting the nCEO pin of each device to the nCE pin of the subsequent device. The nCE pin of the first device is connected to GND, while its nCEO pin is connected to the nCE of the next device in the chain. The nCE input of the last device comes from the previous device, while its nCEO pin is left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. Because all CONF_DONE pins are tied together, all devices in the chain initialize and enter user mode at the same time.

In addition, because the nSTATUS pins are tied together, the entire chain halts configuration if any device detects an error. The **Auto-restart configuration after error** option does not affect the configuration cycle because you must manually restart configuration in the Quartus II software when an error occurs.

Figure 9–15 shows how to configure multiple Arria II devices with a download cable.

Figure 9–15. Multi-Device PS Configuration Using a USB-Blaster, EthernetBlaster, EthernetBlaster II, or ByteBlaster II Cable



Notes to Figure 9–15:

(1) Connect the pull-up resistor to the same supply voltage, V_{CCIO} for Arria II GX devices or V_{CCPGM} for Arria II GZ devices as the USB-Blaster, EthernetBlaster, EthernetBlaster II, or ByteBlaster II cable.

(2) You only need the pull-up resistors on DATA0 and DCLK if the download cable is the only configuration scheme used on your board. This ensures that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, you do not need the pull-up resistors on DATA0 and DCLK.

(3) The MSEL pin settings vary for different configuration voltage standards and POR delays. To connect MSEL[3..0]for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect MSEL[2..0] for an Arria II GZ device, refer to Table 9–7 on page 9–10.

(4) In the USB-Blaster and ByteBlaster II cables, this pin is connected to nCE when it is used for AS programming; otherwise, it is a no connect.

For more information about how to use the USB-Blaster, ByteBlaster II, EthernetBlaster, or EthernetBlaster II cables, refer to the following user guides:

- *ByteBlaster II Download Cable User Guide*
- *EthernetBlaster Communications Cable User Guide*
- *EthernetBlaster II Communications Cable User Guide*
- *USB-Blaster Download Cable User Guide*

JTAG Configuration

JTAG has developed a specification for boundary-scan testing. This boundary-scan test (BST) architecture offers the capability to efficiently test components on PCBs with tight lead spacing. The BST architecture can test pin connections without using physical test probes and capture functional data while a device is operating normally. You can also use JTAG circuitry to shift configuration data into the device. The Quartus II software automatically generates **.sof**s that you can use for JTAG configuration with a download cable in the Quartus II software programmer.

For more information about JTAG boundary-scan testing and commands available using Arria II devices, refer to the following documents:

- *JTAG Boundary-Scan Testing in Arria II Devices* chapter
- *Programming Support for Jam STAPL Language*

Arria II devices are designed such that JTAG instructions have precedence over any device configuration modes. Therefore, JTAG configuration can take place without waiting for other configuration modes to complete. For example, if you attempt JTAG configuration of Arria II devices during PS configuration, PS configuration is terminated and JTAG configuration begins.

You cannot use the Arria II decompression or design security features if you are configuring your Arria II device using JTAG-based configuration.

A device operating in JTAG mode uses four required pins, TDI, TDO, TMS, and TCK, and one optional pin, TRST. The TCK pin has an internal weak pull-down resistor, while the TDI, TMS, and TRST pins have weak internal pull-up resistors (typically 25 kΩ). All the JTAG pins are powered by the V_{CCIO} power supply of I/O bank 8C for Arria II GX devices and 2.5-V/3.0-V V_{CCPD} power supply for Arria II GZ devices. All the JTAG pins support only the LVTTL I/O standard.

All user I/O pins are tri-stated during JTAG configuration. Table 9–13 lists the function of each JTAG pin.

For more information about how to connect a JTAG chain with multiple voltages across the devices in the chain, refer to the *JTAG Boundary-Scan Testing in Arria II Devices* chapter.

Table 9–13. JTAG Pins Signals (Part 1 of 2)

Pin Name	Pin Type	Description
TDI	Test data input	Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by connecting this pin to logic high.
TDO	Test data output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by leaving this pin unconnected.

Table 9–13. JTAG Pins Signals (Part 2 of 2)

Pin Name	Pin Type	Description
TMS	Test mode select	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. TMS is evaluated on the rising edge of TCK. Therefore, you must set up TMS before the rising edge of TCK. Transitions within the state machine occur on the falling edge of TCK after the signal is applied to TMS. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by connecting this pin to logic high.
TCK	Test clock input	Clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by connecting TCK to GND.
TRST *(1)*	Test reset input (optional)	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to the IEEE Std. 1149.1 standard. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by connecting the TRST pin to GND. One kΩ pull-up resistor to V_{CCPD} if you do not use the TRST pin.

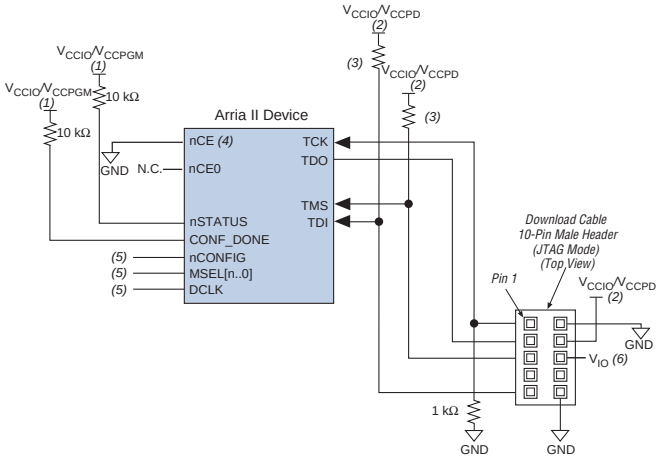
Note to Table 9–13:

(1) The TRST pin is only available for Arria II GZ devices.

During JTAG configuration, you can download data to the device on the PCB through the USB-Blaster, ByteBlaster II, EthernetBlaster, or EthernetBlaster II download cable.

Figure 9–16 shows the JTAG configuration of a single Arria II device.

Figure 9–16. JTAG Configuration of a Single Device Using a Download Cable



Notes to Figure 9–16:

(1) Connect the pull-up resistors to the V_{CCIO} power supply of I/O bank 3C for Arria II GX devices and to V_{CCPGM} at a 3.0-V power supply for Arria II GZ devices.

(2) Connect the pull-up resistor to the same supply voltage, V_{CCIO} for Arria II GX devices or V_{CCPD} for Arria II GZ devices as the USB-Blaster, ByteBlaster II, EthernetBlaster, or EthernetBlaster II cable.

(3) The resistor value can vary from 1 KΩ to 10 KΩ

(4) You must connect nCE to GND or drive it low for successful JTAG configuration.

(5) Connect the nCONFIG and MSEL pins to support a non-JTAG configuration scheme. If you only use the JTAG configuration, connect nCONFIG to V_{CCIO} for Arria II GX device, V_{CCPGM} for Arria II GZ device, and MSEL to GND. Pull DCLK either high or low, whichever is convenient on your board.

(6) In the USB-Blaster and ByteBlaster II cables, this pin is a no connect.

To configure a single device in a JTAG chain, the programming software places all other devices in bypass mode. In bypass mode, devices pass programming data from the TDI pin to the TDO pin through a single bypass register without being affected internally. This scheme enables the programming software to program or verify the target device. Configuration data driven into the device appears on the TDO pin one clock cycle later.

The Quartus II software verifies successful JTAG configuration after completion. At the end of configuration, the software checks the state of CONF_DONE through the JTAG port. When the Quartus II software generates a Jam™ file (.**jam)** for a multi-device chain, it contains instructions so that all the devices in the chain are initialized at the same time. If CONF_DONE is not high, the Quartus II software indicates that configuration has failed. If CONF_DONE is high, the software indicates that configuration was successful. After the configuration bitstream is transmitted serially using the JTAG TDI port, the TCK port is clocked an additional 1,094 cycles to perform device initialization.

Arria II devices have dedicated JTAG pins that always function as JTAG pins. Not only can you perform JTAG testing on Arria II devices before and after, but also during configuration. While other device families do not support JTAG testing during configuration, Arria II devices support the bypass, ID code, and sample instructions during configuration without interrupting configuration. All other JTAG instructions may only be issued by first interrupting configuration and reprogramming I/O pins using the CONFIG_IO instruction.

The CONFIG_IO instruction allows I/O buffers to be configured using the JTAG port and when issued, interrupts configuration. This instruction allows you to perform board-level testing prior to configuring the Arria II device or waiting for a configuration device to complete configuration. After configuration is interrupted and JTAG testing is complete, you must reconfigure the part using JTAG (PULSE_CONFIG instruction) or by pulsing nCONFIG low.

The chip-wide reset (DEV_CLRn) and chip-wide output enable (DEV_OE) pins on Arria II devices do not affect JTAG boundary-scan or programming operations. Toggling these pins does not affect JTAG operations (other than the usual boundary-scan operation).

When designing a board for JTAG configuration of Arria II devices, consider the dedicated configuration pins. Table 9–14 lists how these pins are connected during JTAG configuration.

Table 9–14. Dedicated Configuration Pin Connections During JTAG Configuration (Part 1 of 2)

Signal	Description
nCE	On all Arria II devices in the chain, nCE must be driven low by connecting it to GND ground, pulling it low using a resistor, or driving it by some control circuitry. For devices that are also in multi-device FPP, AS, or PS configuration chains, the nCE pins must be connected to GND during JTAG configuration or JTAG must be configured in the same order as the configuration chain.
nCEO	On all Arria II devices in the chain, you can leave nCEO floating or connected to nCE of the next device.
MSEL	Do not leave these pins floating. These pins support whichever non-JTAG configuration is used in production. If you only use JTAG configuration, tie these pins to GND.
nCONFIG	Driven high by connecting to V_{CCIO} or V_{CCPGM}, pulling up using a resistor, or driven high by some control circuitry.

Table 9–14. Dedicated Configuration Pin Connections During JTAG Configuration (Part 2 of 2)

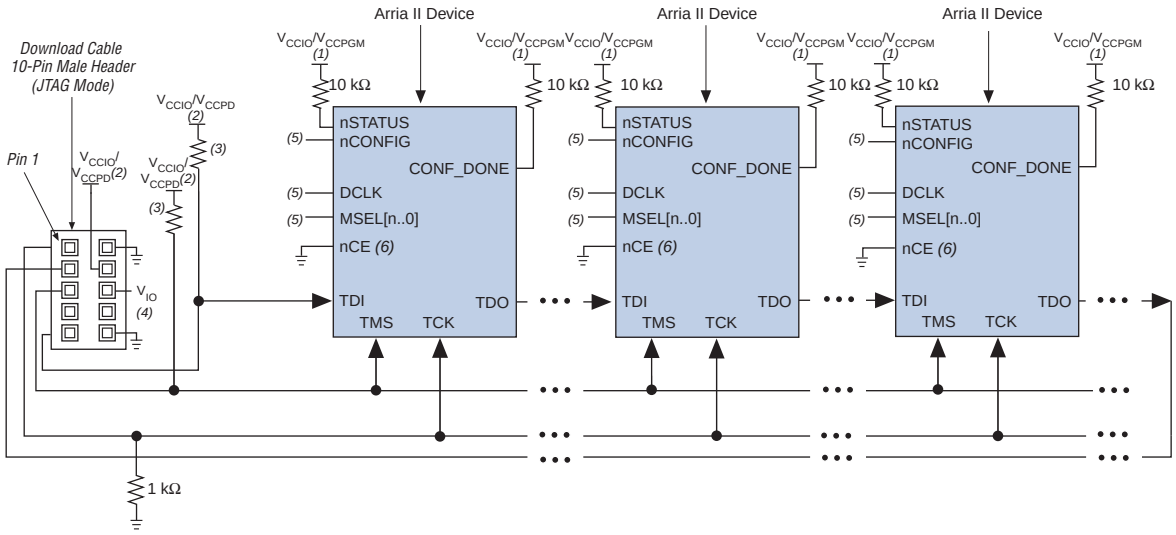
Signal	Description
nSTATUS	Pull to V_{CCIO} or V_{CCPGM} using a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each nSTATUS pin must be pulled up to V_{CCIO} or V_{CCPGM} individually.
CONF_DONE	Pull to V_{CCIO} or V_{CCPGM} using a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each CONF_DONE pin must be pulled up to V_{CCIO} or V_{CCPGM} individually. CONF_DONE going high at the end of JTAG configuration indicates successful configuration.
DCLK	Do not leave DCLK floating. Drive low or high, whichever is more convenient on your board.

When programming a JTAG device chain, one JTAG-compatible header is connected to several devices. The number of devices in the JTAG chain is limited only by the drive capability of the download cable. When four or more devices are connected in a JTAG chain, Altera recommends buffering the TCK, TDI, and TMS pins with an on-board buffer.

JTAG-chain device programming is ideal when the system contains multiple devices or when testing your system using JTAG BST circuitry.

Figure 9–17 shows a multi-device JTAG configuration when using a download cable.

Figure 9–17. JTAG Configuration of Multiple Devices Using a Download Cable



Notes to Figure 9–17:

(1) Connect the pull-up resistors to the V_{CCIO} power supply of I/O bank 3C for Arria II GX devices and to V_{CCPGM} at a 3.0-V power supply for Arria II GZ devices.

(2) You must connect the pull-up resistor to the same supply voltage, V_{CCIO} for Arria II GX devices or V_{CCPD} for Arria II GZ devices as the USB-Blaster, ByteBlaster II, EthernetBlaster, or EthernetBlaster II cable.

(3) The resistor value can vary from 1 KΩ to 10 KΩ

(4) In the USB-Blaster and ByteBlaster II cables, pin 6 is a no connect.

(5) You must connect the nCONFIG and MSEL pins to support a non-JTAG configuration scheme. If you only use JTAG configuration, connect nCONFIG to the V_{CCIO} for Arria II GX device, V_{CCPGM} for Arria II GZ device, and MSEL to GND. Pull DCLK either high or low, whichever is convenient on your board.

(6) You must connect nCE to GND or drive it low for successful JTAG configuration.

You must connect the nCE pin to GND or drive it low during JTAG configuration. In multi-device FPP, AS, and PS configuration chains, the nCE pin of the first device is connected to GND, while its nCEO pin is connected to nCE of the next device in the chain. The nCE input of the last device comes from the previous device, while its nCEO pin is left floating. In addition, the CONF_DONE and nSTATUS signals are all shared in multi-device FPP, AS, or PS configuration chains so the devices can enter user mode at the same time after configuration is complete. When the CONF_DONE and nSTATUS signals are shared among all the devices, you must configure every device when JTAG configuration is performed.

If you only use JTAG configuration, Altera recommends connecting the circuitry as shown in Figure 9–17, where each of the CONF_DONE and nSTATUS signals are isolated to enable each device to enter user mode individually.

After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the nCE pin of the second device, which prompts the second device to begin configuration. Therefore, if these devices are also in a JTAG chain, ensure the nCE pins are connected to GND during JTAG configuration or that the devices are JTAG configured in the same order as the configuration chain. As long as the devices are JTAG configured in the same order as the multi-device configuration chain, the nCEO of the previous device drives the nCE of the next device low when it has successfully been JTAG configured.

You can place other Altera devices that have JTAG support in the same JTAG chain for device programming and configuration.

JTAG configuration support is enhanced and allows more than 17 Arria II devices to be cascaded in a JTAG chain.

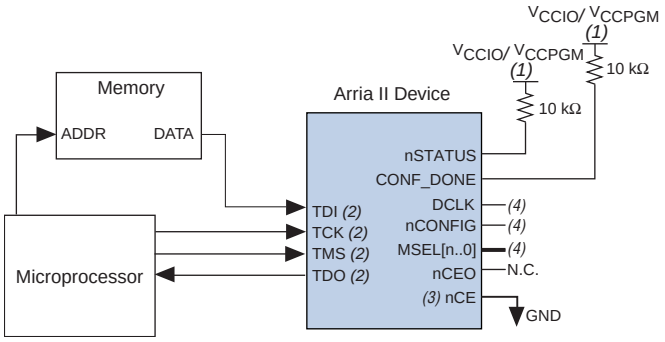
For more information about configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera Device Chains* chapter in volume 2 of the *Configuration Handbook*.

You can configure Arria II devices using multiple configuration schemes on the same board. Combining JTAG configuration with a PS or AS configuration on your board is useful in the prototyping environment because it allows multiple methods to configure your FPGA.

For more information about combining JTAG configuration with other configuration schemes, refer to the *Combining Different Configuration Schemes* chapter in volume 2 of the *Configuration Handbook*.

Figure 9–18 shows a JTAG configuration of an Arria II device using a microprocessor.

Figure 9–18. JTAG Configuration of a Single Device Using a Microprocessor



Notes to Figure 9–18:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for all Arria II devices in the chain. The V_{CCIO} power supply for Arria II GX devices or the V_{CCPGM} power supply for Arria II GZ devices must be high enough to meet the V_{IH} specification of the I/O on the device.

(2) To drive the JTAG pins, the microprocessor must use the same I/O standard as V_{CCIO} for Arria II GX devices or V_{CCPD} for Arria II GZ devices.

(3) You must connect nCE to GND or drive it low for successful JTAG configuration.

(4) Connect the nCONFIG and MSEL pins to support a non-JTAG configuration scheme. If you use only the JTAG configuration, connect nCONFIG to the V_{CCIO} for Arria II GX device, V_{CCPGM} for Arria II GZ device, and MSEL to GND. Pull DCLK either high or low, whichever is convenient on your board. Arria II GX devices use MSEL[3..0] pins while Arria II GZ devices use MSEL[2..0] pins.

Jam STAPL

Jam standard test and programming language (STAPL), JEDEC standard JESD-71, is a standard file format for in-system programmability (ISP) purposes. Jam STAPL supports programming or configuration of programmable devices and testing of electronic systems, using the IEEE 1149.1 JTAG interface. Jam STAPL is a freely licensed open standard.

The Jam Player provides an interface for manipulating the IEEE Std. 1149.1 JTAG TAP state machine.

For more information about JTAG and Jam STAPL in embedded environments, refer to *AN 425: Using Command-Line Jam STAPL Solution for Device Programming*. To download the Jam Player, visit the Altera website.

Device Configuration Pins

Table 9–15 through Table 9–18 list the connections and functionality of all the configuration-related pins on the Arria II devices.

Table 9–15 lists the Arria II configuration pins and their power supply.

Table 9–15. Configuration Pin Summary for Arria II Devices

Description	Input/Output	Dedicated	Powered By *(1)*	Configuration Mode
TDI	Input	Yes	V_{CCPD}/V_{CCIO}	JTAG
TMS	Input	Yes	V_{CCPD}/V_{CCIO}	JTAG
TCK	Input	Yes	V_{CCPD}/V_{CCIO}	JTAG
TRST	Input	Yes	V_{CCPD}/V_{CCIO}	JTAG
TDO	Output	Yes	V_{CCPD}/V_{CCIO}	JTAG
CRC_ERROR	Output	—	Pull-up	Optional, all modes
DATA0	Input	—	V_{CCPGM}/V_{CCIO} *(2)*	All modes except JTAG
DATA[7..1]	Input	—	V_{CCPGM}/V_{CCIO} *(2)*	FPP
INIT_DONE	Output	—	Pull-up	Optional, all modes
CLKUSR	Input	—	V_{CCPGM}/V_{CCIO} *(2)*	Optional
nSTATUS	Bidirectional	Yes	V_{CCPGM}/Pull-up	All modes
nCE	Input	Yes	V_{CCPGM}/V_{CCIO}	All modes
CONF_DONE	Bidirectional	Yes	V_{CCPGM}/Pull-up	All modes
nCONFIG	Input	Yes	V_{CCPGM}/V_{CCIO}	All modes
ASDO	Output	Yes	V_{CCPGM}/V_{CCIO}	AS
nCSO	Output	Yes	V_{CCPGM}/V_{CCIO}	AS
DCLK	Input	Yes	V_{CCPGM}/V_{CCIO}	PS, FPP
	Output	Yes	V_{CCPGM}/V_{CCIO}	AS
nIO_PULLUP	Input	Yes	V_{CC} *(3)*	All modes
nCEO	Output	—	V_{CCPGM}/Pull-up	All modes
MSEL[2..0] *(4)*	Input	Yes	V_{CCIO} *(6)*	All modes
MSEL[3..0] *(5)*	Input	Yes	V_{CCPD}	All modes

Notes to Table 9–15:

(1) Arria II GX devices use V_{CCIO} while Arria II GZ devices use V_{CCPD}.

(2) For Arria II GZ devices, these pins are powered up by V_{CCPGM} during configuration and V_{CCIO} if they are used as a regular I/O in user mode.

(3) Although the nIO_PULLUP is powered up by V_{CC}, Altera recommends connecting this pin to V_{CCPGM} for Arria II GZ devices, V_{CCIO} for Arria II GX devices, or GND directly without using a pull-up or pull-down resistor.

(4) Arria II GZ devices use a MSEL[2..0] configuration scheme.

(5) Arria II GX devices use a MSEL[3..0] configuration scheme.

(6) Although MSEL[2..0], PORSEL, and nIO_PULLUP are powered by V_{CC}, Altera recommends connecting these pins to V_{CCPGM} or GND directly without using a pull-up or pull-down resistor.

Table 9–16 lists the dedicated configuration pins. You must connect these pins properly on your board for successful configuration. Some of these pins may not be required for your configuration schemes.

Table 9–16. Dedicated Configuration Pins on the Arria II Device (Part 1 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
VCCPD	N/A	All	Power *(1)*	Dedicated power pin. Use this pin to power the I/O pre-drivers, the HSTL/SSTL input buffers, and the MSEL[3..0] pins. You must connect V_{CCPD} according to the I/O standard used in the same bank: ■ For 3.3-V I/O standards, connect V_{CCPD} to 3.3 V ■ For 3.0-V I/O standards, connect V_{CCPD} to 3.0 V ■ For 2.5-V and below I/O standards, connect V_{CCPD} to 2.5 V V_{CCPD} must ramp up from 0 V to 2.5, 3.0, or 3.3 V in 100 ms (for standard POR) or 4 ms (for fast POR). If V_{CCPD} is not ramped up in this specified time, your Arria II device is not successfully configured.
nIO_PULLUP	N/A	All	Input	Dedicated input that chooses whether the internal pull-up resistors on the user I/O pins and dual-purpose I/O pins (DATA[7..0], CLKUSR, INIT_DONE, DEV_OE, and DEV_CLRn) are on or off before and during configuration. A logic high turns off the weak internal pull-up resistors; a logic low turns them on. The nIO-PULLUP input buffer is powered by V_{CC} and has an internal 5-kΩ pull-down resistor that is always active. You can tie the nIO-PULLUP directly to the V_{CCPGM} power supply for Arria II GZ devices and the V_{CCIO} power supply for Arria II GX devices, or GND.
MSEL[2..0]	N/A	All	Input	Three-bit configuration input that sets the Arria II GZ device configuration scheme. For more information about the appropriate connections, refer to Table 9–7 on page 9–10. You must hardwire these pins to V_{CCPGM} or GND. The MSEL[2..0] pins have internal 5-kΩ pull-down resistors that are always active.
MSEL[3..0]	N/A	All	Input	Four-bit configuration input that sets the Arria II GX device configuration scheme. For more information about the appropriate connections, refer to Table 9–6 on page 9–9. You must hardwire these pins to V_{CCPD} or GND. The MSEL[3..0] pins have internal 5-kΩ pull-down resistors that are always active.
nCONFIG	N/A	All	Input	Configuration control input. Pulling this pin low during user-mode causes the device to lose its configuration data, enter a reset state, and tri-state all I/O pins. Returning this pin to a logic-high level starts a reconfiguration. Configuration is possible only if this pin is high, except in JTAG programming mode, when nCONFIG is ignored.

Table 9–16. Dedicated Configuration Pins on the Arria II Device (Part 2 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nSTATUS	N/A	All	Bidirectional open-drain	The device drives nSTATUS low immediately after power-up and releases it after the POR time. During user mode and regular configuration, this pin is pulled high by an external 10-kΩ resistor. This pin, when driven low by the Arria II device, indicates that the device has encountered an error during configuration. ■ Status output—If an error occurs during configuration, nSTATUS is pulled low by the target device. ■ Status input—If an external source drives the nSTATUS pin low during configuration or initialization, the target device enters an error state. Driving nSTATUS low after configuration and initialization does not affect the configured device. If you use a configuration device, driving nSTATUS low causes the configuration device to attempt to configure the device, but because the device ignores transitions on nSTATUS in user mode, the device does not reconfigure. To begin a reconfiguration, nCONFIG must be pulled low. If V_{CCIO} for Arria II GX devices or V_{CCPGM} for Arria II GZ devices are not fully powered up, the following could occur: ■ V_{CCIO}/V_{CCPGM} is powered high enough for the nSTATUS buffer to function properly and nSTATUS is driven low. When V_{CCIO}/V_{CCPGM} is ramped up, POR trips and nSTATUS is released after POR expires. ■ V_{CCIO}/V_{CCPGM} is not powered high enough for the nSTATUS buffer to function properly. In this situation, nSTATUS might appear logic high, triggering a configuration attempt that fails because POR did not yet trip. When V_{CCPD} is powered up, nSTATUS is pulled low because POR did not yet trip. When POR trips after V_{CCIO}/V_{CCPGM} is powered up, nSTATUS is released and pulled high. At that point, reconfiguration is triggered and the device is configured.
CONF_DONE	N/A	All	Bidirectional open-drain	Status output. The target device drives the CONF_DONE pin low before and during configuration. After all configuration data is received without error and the initialization cycle starts, the target device releases CONF_DONE. Status input. After all data is received and CONF_DONE goes high, the target device initializes and enters user mode. The CONF_DONE pin must have an external 10-kΩ pull-up resistor for the device to initialize. Driving CONF_DONE low after configuration and initialization does not affect the configured device.

Table 9–16. Dedicated Configuration Pins on the Arria II Device (Part 3 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nCE	N/A	All	Input	Active-low chip enable. The nCE pin activates the device with a low signal to allow configuration. The nCE pin must be held low during configuration, initialization, and user mode. In single device configuration, it must be tied low. In multi-device configuration, nCE of the first device is tied low while its nCEO pin is connected to nCE of the next device in the chain. The nCE pin must also be held low for successful JTAG programming of the device.
nCEO	I/O	All	Output open-drain	Output that drives low when device configuration is complete. In a single-device configuration, this pin is left floating. In a multi-device configuration, this pin feeds the next device's nCE pin and is pulled high by an external 10-kΩ resistor. The nCEO of the last device in the chain is left floating. The nCEO pin is powered by V_{CCIO} for Arria II GX devices and V_{CCPGM} for Arria II GZ devices. After configuration, nCEO is available as user I/O pins. The state of the nCEO pin depends on the **Dual-Purpose Pin** settings.
ASDO *(2)*	N/A	AS	Output	Control signal from the Arria II device to the serial configuration device in AS mode used to read out configuration data. In AS mode, ASDO has an internal pull-up resistor that is always active.
nCSO *(2)*	N/A	AS	Output	Output control signal from the Arria II device to the serial configuration device in AS mode that enables the configuration device. In AS mode, nCSO has an internal pull-up resistor that is always active.
DCLK *(2)*	N/A	Synchronous configuration schemes (PS, FPP, AS)	Input (PS, FPP) Output (AS)	In PS and FPP configurations, DCLK is the clock input used to clock data from an external source into the target device. Data is latched into the device on the rising edge of DCLK. In AS mode, DCLK is an output from the Arria II device that provides timing for the configuration interface. In AS mode, DCLK has an internal pull-up resistor (typically 25 kΩ) that is always active. After configuration, this pin by default is driven into an inactive state. In schemes that use a control host, DCLK must be driven either high or low, whichever is more convenient. Toggling this pin after configuration does not affect the configured device.

Table 9–16. Dedicated Configuration Pins on the Arria II Device (Part 4 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
DATA0 *(2)*	N/A	PS, FPP, AS	Input	Data input. In serial configuration modes, bit-wide configuration data is presented to the target device on the DATA0 pin. In AS mode, DATA0 has an internal pull-up resistor that is always active. For Arria II GX devices, DATA0 is a dedicated pin that is used for both PS and AS configuration modes and is not available as a user I/O pin after configuration. For Arria II GZ devices, after PS or FPP configuration, DATA0 is available as a user I/O pin. The state of this pin depends on the **Dual-Purpose Pin** settings.
DATA[7..1]	I/O	Parallel configuration schemes (FPP)	Inputs	Data inputs. Byte-wide configuration data is presented to the target device on DATA[7..0]. In serial configuration schemes, they function as user I/O pins during configuration, which means they are tri-stated. After FPP configuration, DATA[7..1] are available as user I/O pins. The state of these pin depends on the **Dual-Purpose Pin** settings.

Notes to Table 9–16:

(1) Arria II GZ devices do not support the 3.3-V I/O standard.

(2) To tri-state the AS configuration pins in user mode, turn on the **Enable input tri-state on active configuration pins in user mode** option from the **Device and Pin Options** dialog box in the **Configuration** tab. This tri-states the DCLK, DATA0, nCSO, and ASDO pins.

Table 9–17 lists the optional configuration pins. If these optional configuration pins are not enabled in the Quartus II software, they are available as general-purpose user I/O pins. Therefore, during configuration, these pins function as user I/O pins and are tri-stated with weak pull-up resistors.

Table 9–17. Optional Configuration Pins

Pin Name	User Mode	Pin Type	Description
CLKUSR	N/A if option is on. I/O if option is off.	Input	Optional user-supplied clock input synchronizes the initialization of one or more devices. Enable this pin by turning on the **Enable user-supplied start-up clock** (**CLKUSR**) option in the Quartus II software.
INIT_DONE	N/A if option is on. I/O if option is off.	Output open-drain	Use as a status pin to indicate when the device has initialized and is in user mode. When nCONFIG is low and during the beginning of configuration, the INIT_DONE pin is tri-stated and pulled high due to an external 10-kΩ pull-up resistor. After the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high and the device enters user mode. Thus, the monitoring circuitry must be able to detect a low-to-high transition. Enable this pin by turning on the **Enable INIT_DONE output** option in the Quartus II software.
DEV_OE	N/A if option is on. I/O if option is off.	Input	Optional pin that allows you to override all tri-states on the device. When this pin is driven low, all I/O pins are tri-stated. When this pin is driven high, all I/O pins behave as programmed. Enable this pin by turning on the **Enable device-wide output enable** (**DEV_OE**) option in the Quartus II software.
DEV_CLRn	N/A if option is on. I/O if option is off.	Input	Optional pin that allows you to override all clears on all device registers. When this pin is driven low, all registers are cleared. When this pin is driven high, all registers behave as programmed. Enable this pin by turning on the **Enable device-wide reset** (**DEV_CLRn**) option in the Quartus II software.

Table 9–18 lists the dedicated JTAG pins. JTAG pins must be kept stable before and during configuration to prevent accidental loading of JTAG instructions. The TDI, TMS, and TRST pins have weak internal pull-up resistors; the TCK pin has a weak internal pull-down resistor (typically 25 kΩ). If you plan to use the SignalTap™ embedded logic array analyzer, you must connect the JTAG pins of the Arria II device to a JTAG header on your board.

Table 9–18. Dedicated JTAG Pins

Pin Name	User Mode	Pin Type	Description
TDI	N/A	Test data input	Serial input pin for instructions as well as test and programming data. Data is shifted on the rising edge of TCK. The TDI pin is powered by the V_{CCIO} power supply for Arria II GX devices and the V_{CCPD} power supply for Arria II GZ devices. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by connecting this pin to logic high.
TDO	N/A	Test data output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. The TDO pin is powered up by the V_{CCPD}/V_{CCIO} power supply. For more information about connecting a JTAG chain with multiple voltages across the devices in the chain, refer to the *JTAG Boundary-Scan Testing in Arria II Devices* chapter. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by leaving this pin unconnected.
TMS	N/A	Test mode select	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. TMS is evaluated on the rising edge of TCK. Therefore, you must set up TMS before the rising edge of TCK. Transitions in the state machine occur on the falling edge of TCK after the signal is applied to TMS. The TMS pin is powered by the V_{CCPD}/V_{CCIO} power supply. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by connecting this pin to logic high.
TCK	N/A	Test clock input	Clock input to the BST circuitry. Some operations occur at the rising edge while others occur at the falling edge. The TCK pin is powered by the V_{CCPD}/V_{CCIO} power supply. It is expected that the clock input waveform have a nominal 50% duty cycle. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by connecting TCK to GND.
TRST *(1)*	N/A	Test reset input (optional)	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to the IEEE Std. 1149.1 standard. The TRST pin is powered by the V_{CCPD} power supply for Arria II GZ devices. Hold TMS at one or keep TCK static while TRST is changed from 0 to 1. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by connecting the TRST pin to GND. You need one kΩ pull-up resistor to V_{CCPD} if you do not use the TRST pin.

Note to Table 9–18:

(1) The TRST pin is only available for Arria II GZ devices.

Configuration Data Decompression

Arria II devices support configuration data decompression, which saves configuration memory space and time. This feature allows you to store compressed configuration data in configuration devices or other memory and transmit this compressed bitstream to Arria II devices. During configuration, the Arria II device decompresses the bitstream in real time and programs its SRAM cells.

Data indicates that compression typically reduces the configuration bitstream size by 35 to 55% based on the designs used.

Arria II devices support decompression in the FPP (when using a MAX II device or microprocessor + flash), AS or fast AS, and PS configuration schemes. The Arria II device decompression feature is not available in the JTAG configuration scheme.

When using FPP mode, the intelligent host must provide a DCLK that is x4 the data rate. Therefore, the configuration data must be valid for four DCLK cycles.

In PS mode, use the Arria II decompression feature because sending compressed configuration data reduces configuration time.

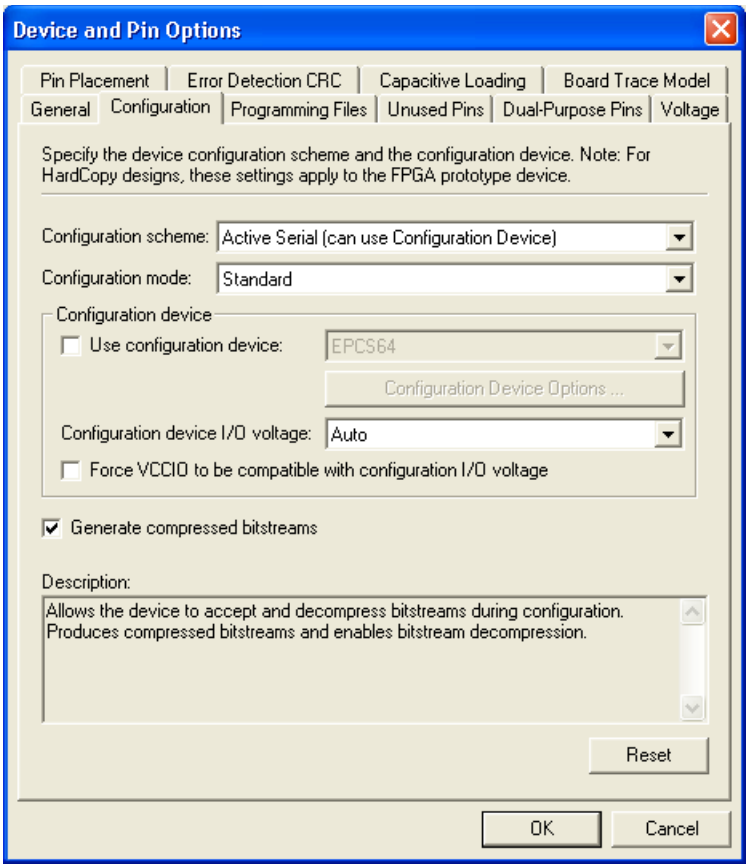
When you enable compression, the Quartus II software generates configuration files with compressed configuration data. This compressed file reduces the storage requirements in the configuration device or flash memory and decreases the time needed to transmit the bitstream to the Arria II device. The time required by an Arria II device to decompress a configuration file is less than the time needed to transmit the configuration data to the device.

There are two ways to enable compression for Arria II bitstreams—before design compilation (in the Compiler Settings menu) and after design compilation (in the **Convert Programming Files** window).

To enable compression in the project's Compiler Settings menu, follow these steps:

1. On the Assignments menu, click **Device**. The **Settings** dialog box appears.
2. After selecting your Arria II device, open the **Device and Pin Options** window.
3. In the **Configuration** settings tab, turn on **Generate compressed bitstreams** (as shown in Figure 9–19).

Figure 9–19. Enabling Compression for Arria II Bitstreams in Compiler Settings



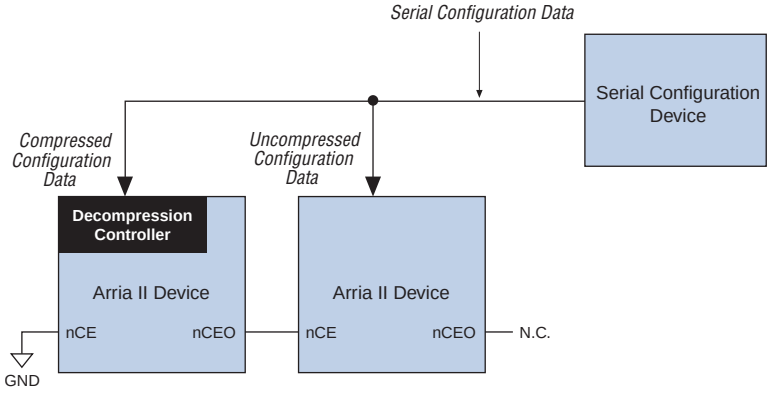
You can also enable compression when creating programming files from the **Convert Programming Files** window. To do this, follow these steps:

1. On the File menu, click **Convert Programming Files**.
2. Select the programming file type (**.pof**, **.sram**, **.hex**, **.rbf**, or **.ttf**).
3. For **.pof** output files, select a configuration device.
4. In the **Input files to convert** box, select **SOF Data**.
5. Select **Add File** and add an Arria II device **.sof**.
6. Select the name of the file you added to the **SOF Data** area and click **Properties**.
7. Check the **Compression** check box.

When multiple Arria II devices are cascaded, you can selectively enable the compression feature for each device in the chain if you are using a serial configuration scheme. Figure 9–20 shows a chain of two Arria II devices. The first Arria II device has compression enabled; therefore, receives a compressed bitstream from the configuration device. The second Arria II device has the compression feature disabled and receives uncompressed data.

In a multi-device FPP configuration chain (with a MAX II device or microprocessor + flash), all Arria II devices in the chain must either enable or disable the decompression feature. You cannot selectively enable the compression feature for each device in the chain because of the DATA and DCLK relationship.

Figure 9–20. Compressed and Uncompressed Serial Configuration Data in the Same Configuration File



You can generate programming files for this setup by clicking **Convert Programming Files** on the File menu in the Quartus II software.

Remote System Upgrades

This section describes the functionality and implementation of the dedicated remote system upgrade circuitry. It also defines several concepts related to remote system upgrades, including factory configuration, application configuration, remote update mode, and user watchdog timer. Additionally, this section provides design guidelines for implementing remote system upgrades with the supported configuration schemes.

System designers sometimes face challenges such as shortened design cycles, evolving standards, and system deployments in remote locations. Arria II devices help overcome these challenges with their inherent reprogrammability and dedicated circuitry to perform remote system upgrades. Remote system upgrades help deliver feature enhancements and bug fixes without costly recalls, reduce time-to-market, extend product life, and help to avoid system downtime.

Arria II devices feature dedicated remote system upgrade circuitry. Soft logic (either the Nios® II embedded processor or user logic) implemented in an Arria II device can download a new configuration image from a remote location, store it in configuration memory, and direct the dedicated remote system upgrade circuitry to start a reconfiguration cycle. The dedicated circuitry performs error detection during and after the configuration process, recovers from any error condition by reverting back to a safe configuration image, and provides error status information.

Remote system upgrades are supported in AS configuration schemes for Arria II GX devices and in fast AS configuration schemes for Arria II GZ devices. You can also implement remote system upgrades in conjunction with advanced Arria II features such as real-time decompression of configuration data and design security using the advanced encryption standard (AES) for secure and efficient field upgrades. The largest serial configuration device currently supports 128 MBits of configuration bitstream.

Arria II devices only support remote system upgrade in the single device fast AS configuration scheme. Because the largest serial configuration device currently supports 128 Mbits of configuration bitstream, the remote system upgrade feature is not supported in EP2AGZ300, EP2AGZ350, and larger devices.

The remote system upgrade feature is not supported in a multi-device chain.

Functional Description

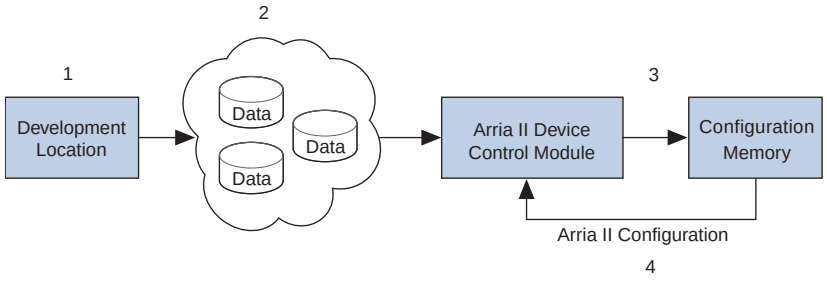
The dedicated remote system upgrade circuitry in Arria II devices manages remote configuration and provides error detection, recovery, and status information. User logic or a Nios II processor implemented in the Arria II device logic array provides access to the remote configuration data source and an interface to the system's configuration memory.

Arria II devices have remote system upgrade processes that involve the following steps:

1. A Nios II processor (or user logic) implemented in the Arria II device logic array receives new configuration data from a remote location. The connection to the remote source uses a communication protocol such as TCP/IP, PCI, user datagram protocol (UDP), UART, or a proprietary interface.
2. The Nios II processor (or user logic) stores this new configuration data in non-volatile configuration memory.
3. The Nios II processor (or user logic) starts a reconfiguration cycle with the new or updated configuration data.
4. The dedicated remote system upgrade circuitry detects and recovers from any error(s) that might occur during or after the reconfiguration cycle and provides error status information to the user design.

Figure 9–21 shows the steps required for performing remote configuration updates. (The numbers in Figure 9–21 coincide with the steps just mentioned.)

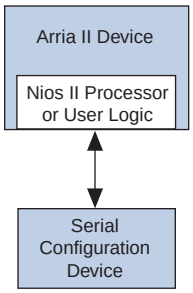
Figure 9–21. Functional Diagram of Arria II Remote System Upgrade



Arria II devices only support remote system upgrade in the single device AS configuration scheme.

Figure 9–22 shows a block diagram for implementing a remote system upgrade with the Arria II configuration scheme.

Figure 9–22. Remote System Upgrade Block Diagram for Arria II Configuration Scheme *(Note 1)*



Note to Figure 9–22:

(1) Arria II GX devices use the AS configuration scheme while Arria II GZ devices use the fast AS configuration scheme.

You must set the mode select `MSEL[3..0]` pins to **AS mode** to use the remote system upgrade feature for Arria II GX devices and the `MSEL[2..0]` pins to **fast AS mode** to use the remote system upgrade feature for Arria II GZ devices.

The `MSEL` pin settings vary for different configuration voltage standards and POR delays. To connect `MSEL[3..0]` for an Arria II GX device, refer to Table 9–6 on page 9–9. To connect `MSEL[2..0]` for an Arria II GZ device, refer to Table 9–7 on page 9–10.

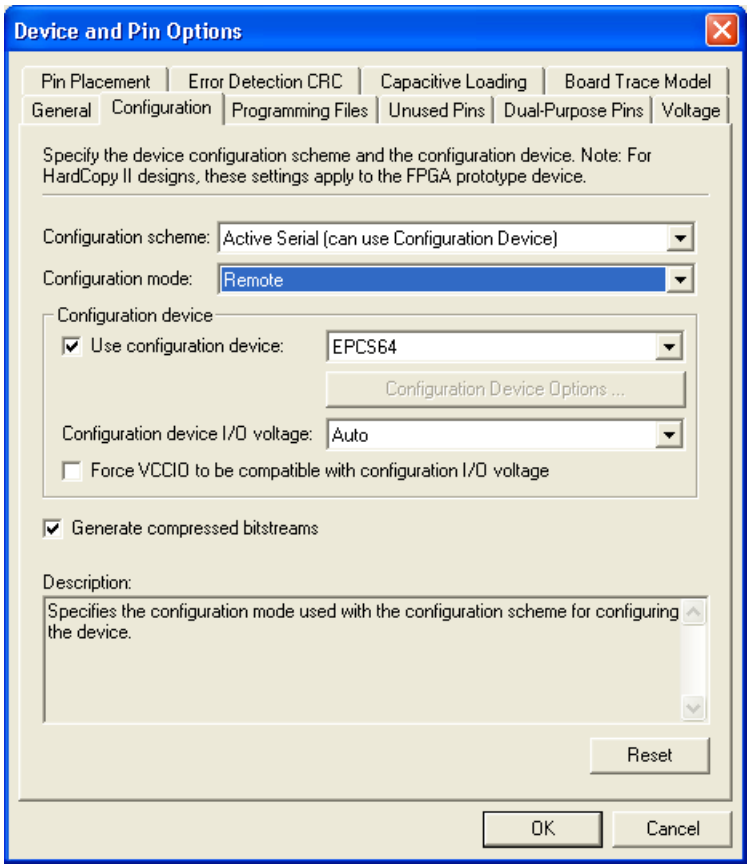
When using AS mode, you must select **remote update mode** in the Quartus II software and insert the ALTREMOTE_UPDATE megafunction to access the circuitry. For more information, refer to "ALTREMOTE_UPDATE Megafunction" on page 9–61.

Enabling Remote Update

You can enable remote update for Arria II devices in the Quartus II software before design compilation (in the Compiler Settings menu). In remote update mode, the **auto-restart configuration after error** option is always enabled. To enable remote update in the project's compiler settings, follow these steps:

1. On the Assignment menu, click **Device**. The **Settings** dialog box appears.
2. Click **Device and Pin Options**. The **Device and Pin Options** dialog box appears.
3. Click the **Configuration** tab.
4. From the **Configuration scheme** list, select **Active Serial** (you can also use **Configuration Device**) (Figure 9–23).
5. From the **Configuration Mode** list, select **Remote** (Figure 9–23).
6. Click **OK**.
7. In the **Settings** dialog box, click **OK**.

Figure 9–23. Enabling Remote Update in the Compiler Settings Menu for Arria II Devices



Configuration Image Types

When performing a remote system upgrade, Arria II device configuration bitstreams are classified as factory configuration images or application configuration images. An image, also referred to as a configuration, is a design loaded into the Arria II device that performs certain user-defined functions.

Each Arria II device in your system requires one factory image or the addition of one or more application images. The factory image is a user-defined fall-back, or safe configuration, and is responsible for administering remote updates in conjunction with the dedicated circuitry. Application images implement user-defined functionality in the target Arria II device. You may include the default application image functionality in the factory image.

A remote system upgrade involves storing a new application configuration image or updating an existing one using the remote communication interface. After an application configuration image is stored or updated remotely, the user design in the Arria II device starts a reconfiguration cycle with the new image. Any errors during or after this cycle are detected by the dedicated remote system upgrade circuitry and cause the device to automatically revert to the factory image. The factory image then performs error processing and recovery. The factory configuration is written to the serial configuration device only once by the system manufacturer and must not be remotely updated. On the other hand, application configurations may be remotely updated in the system. Both images can begin system reconfiguration.

Remote System Upgrade Mode

Remote system upgrade has only one mode of operation—remote update mode. Remote update mode allows you to determine the functionality of your system after power-up and offers several features.

Remote Update Mode

In remote update mode, Arria II devices load the factory configuration image after power up. The user-defined factory configuration determines which application configuration is to be loaded and triggers a reconfiguration cycle. The factory configuration may also contain application logic.

When used with serial configuration devices, remote update mode allows an application configuration to start at any flash sector boundary. For example, this translates to a maximum of 128 sectors in the EPCS64 device and 32 sectors in the EPCS16 device. Altera recommends not using the same page in the serial configuration devices for two images. Additionally, remote update mode features a user watchdog timer that determines the validity of an application configuration.

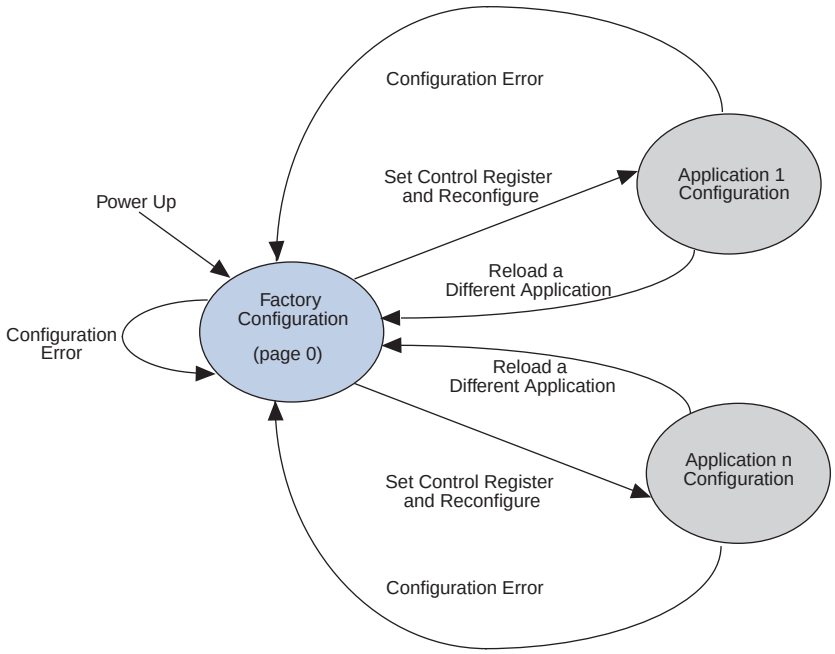
When an Arria II device is first powered up in remote update mode, it loads the factory configuration located at page zero (page registers `PGM[23..0] = 24'b0`). Always store the factory configuration image for your system at page address zero. This corresponds to the start address location 0×000000 in the serial configuration device.

The factory image is user-designed and contains soft logic to:

- Process any errors based on status information from the dedicated remote system upgrade circuitry
- Communicate with the remote host and receive new application configurations and store this new configuration data in the local non-volatile memory device
- Determine which application configuration is to be loaded into the Arria II device
- Enable or disable the user watchdog timer and load its time-out value (optional)
- Instruct the dedicated remote system upgrade circuitry to start a reconfiguration cycle

Figure 9–24 shows the transitions between the factory and the application configurations in remote update mode.

Figure 9–24. Transitions between Configurations in Remote Update Mode



After power up or a configuration error, the factory configuration logic is loaded automatically. The factory configuration also specifies whether to enable the user watchdog timer for the application configuration and if enabled, to include the timer setting information.

The user watchdog timer ensures that the application configuration is valid and functional. The timer must be continually reset in a specific amount of time during user mode operation of an application configuration. Only valid application configurations contain the logic to reset the timer in user mode. This timer reset logic must be part of a user-designed hardware and/or software health monitoring signal that indicates error-free system operation. If the timer is not reset in a specific amount of time; for example, the user application configuration detects a functional problem or if the system hangs, the dedicated circuitry updates the remote system upgrade status register, triggering the loading of the factory configuration.

The user watchdog timer is automatically disabled for factory configurations. For more information about the user watchdog timer, refer to "User Watchdog Timer" on page 9–60.

If there is an error while loading the application configuration, the cause of the reconfiguration is written by the dedicated circuitry to the remote system upgrade status register. Actions that cause the remote system upgrade status register to be written are:

- nSTATUS driven low externally
- Internal CRC error
- User watchdog timer time-out
- A configuration reset (logic array nCONFIG signal or external nCONFIG pin assertion to low)

Arria II devices automatically load the factory configuration located at page address zero. This user-designed factory configuration can read the remote system upgrade status register to determine the reason for the reconfiguration. The factory configuration then takes the appropriate error recovery steps and writes to the remote system upgrade control register to determine the next application configuration to be loaded.

When Arria II devices successfully load the application configuration, they enter into user mode. In user mode, the soft logic (a Nios II processor or state machine and the remote communication interface) assists the Arria II device in determining when a remote system update is arriving. When a remote system update arrives, the soft logic receives the incoming data, writes it to the configuration memory device, and triggers the device to load the factory configuration. The factory configuration reads the remote system upgrade status register and control register, determines the valid application configuration to load, writes the remote system upgrade control register accordingly, and initiates system reconfiguration.

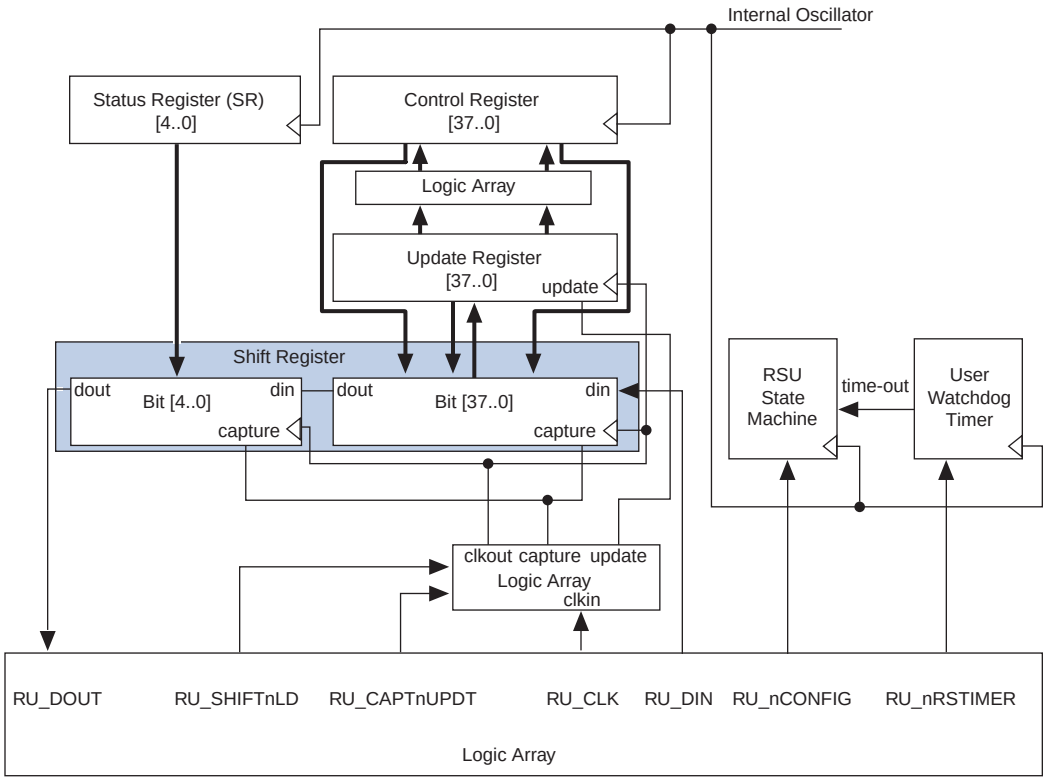
Dedicated Remote System Upgrade Circuitry

This section describes the implementation of the Arria II remote system upgrade dedicated circuitry.

The remote system upgrade circuitry is implemented in hard logic. This dedicated circuitry interfaces with the user-defined factory and application configurations implemented in the Arria II device logic array to provide the complete remote configuration solution. The remote system upgrade circuitry contains the remote system upgrade registers, a watchdog timer, and a state machine that controls those components.

Figure 9–25 shows the datapath of the remote system upgrade block.

Figure 9–25. Remote System Upgrade Circuit Data Path ***(Note 1)***



Note to Figure 9–25:

(1) The RU_DOUT, RU_SHIFTnLD, RU_CAPTnUPDT, RU_CLK, RU_DIN, RU_nCONFIG, and RU_nRSTIMER signals are internally controlled by the ALTREMOTE_UPDATE megafunction.

Remote System Upgrade Registers

The remote system upgrade block contains a series of registers that store the page addresses, watchdog timer settings, and status information. Table 9–19 lists these registers.

Table 9–19. Remote System Upgrade Registers

Register	Description
Shift	This register is accessible by the logic array and allows the update, status, and control registers to be written and sampled by user logic.
Control	This register contains the current page address, user watchdog timer settings, and one bit specifying whether the current configuration is a factory configuration or an application configuration. During a read operation in an application configuration, this register is read into the shift register. When a reconfiguration cycle is initiated, the contents of the update register are written into the control register.
Update	This register contains data similar to that in the control register. However, it can only be updated by the factory configuration by shifting data into the shift register and issuing an update operation. When a reconfiguration cycle is triggered by the factory configuration, the control register is updated with the contents of the update register. During a capture in a factory configuration, this register is read into the shift register.
Status	This register is written to by the remote system upgrade circuitry on every reconfiguration to record the cause of the reconfiguration. This information is used by the factory configuration to determine the appropriate action following a reconfiguration. During a capture cycle, this register is read into the shift register.

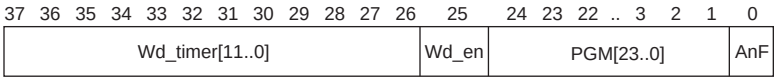
The remote system upgrade control and status registers are clocked by the 10-MHz internal oscillator (the same oscillator that controls the user watchdog timer). However, the remote system upgrade shift and update registers are clocked by the user clock input (RU_CLK).

Remote System Upgrade Control Register

The remote system upgrade control register stores the application configuration page address and user watchdog timer settings. The control register functionality depends on the remote system upgrade mode selection. In remote update mode, the control register page address bits are set to all zeros (24'b0 = 0×000000) at power up to load the factory configuration. A factory configuration in remote update mode has write access to this register.

The control register bit positions are shown in Figure 9–26 and listed in Table 9–20. In the figure, the numbers show the bit position of a setting within a register. For example, bit number 25 is the enable bit for the watchdog timer.

Figure 9–26. Remote System Upgrade Control Register



The application-not-factory (AnF) bit indicates whether the current configuration loaded in the Arria II device is the factory configuration or an application configuration. This bit is set low by the remote system upgrade circuitry when an error condition causes a fall-back to the factory configuration. When the AnF bit is high, the control register access is limited to read operations. When the AnF bit is low, the register allows write operations and disables the watchdog timer.

In remote update mode, the factory configuration design sets this bit high (1'b1) when updating the contents of the update register with the application page address and watchdog timer settings.

Table 9–20 lists the remote system upgrade control register contents.

Table 9–20. Remote System Upgrade Control Register Contents

Control Register Bit	Remote System Upgrade Mode	Value *(1)*	Definition
AnF *(2)*	Remote update	1'b0	Application not factory
PGM[23..0]	Remote update	24'b0×000000	AS configuration start address (StAdd[23..0])
Wd_en	Remote update	1'b0	User watchdog timer enable bit
Wd_timer[11..0]	Remote update	12'b000000000000	User watchdog time-out value (most significant 12 bits of 29-bit count value: {Wd_timer[11..0], 17'b0})

Notes to Table 9–20:

(1) This is the default value of the control register bit.

(2) In remote update mode, the remote configuration block does not update the AnF bit automatically (you can update it manually).

Remote System Upgrade Status Register

The remote system upgrade status register specifies the reconfiguration trigger condition. The various trigger and error conditions include:

- Cyclic redundancy check (CRC) error during application configuration
- nSTATUS assertion by an external device due to an error
- Arria II device logic array triggered a reconfiguration cycle, possibly after downloading a new application configuration image
- External configuration reset (nCONFIG) assertion
- User watchdog timer time-out

The contents of the status register are shown in Figure 9–27 and listed in Table 9–21. The numbers in the figure show the bit positions within a 5-bit register.

Figure 9–27. Remote System Upgrade Status Register

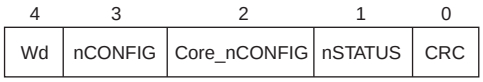


Table 9–21 lists the status register contents for remote system upgrade.

Table 9–21. Remote System Upgrade Status Register Contents

Status Register Bit	Definition	POR Reset Value
CRC (from the configuration)	CRC error caused reconfiguration	1 bit '0'
nSTATUS	nSTATUS caused reconfiguration	1 bit '0'
CORE_nCONFIG *(1)*	Device logic array caused reconfiguration	1 bit '0'
nCONFIG	nCONFIG caused reconfiguration	1 bit '0'
Wd	Watchdog timer caused reconfiguration	1 bit '0'

Note to Table 9–21:

(1) Logic array reconfiguration forces the system to load the application configuration data into the Arria II device. This occurs after the factory configuration specifies the appropriate application configuration page address by updating the update register.

Remote System Upgrade State Machine

The remote system upgrade control and update registers have identical bit definitions, but serve different roles (refer to Figure 9–26 on page 9–57). While both registers can only be updated when the device is loaded with a factory configuration image, the update register writes are controlled by the user logic; the control register writes are controlled by the remote system upgrade state machine.

In factory configurations, the user logic sends the AnF bit (set high), the page address, and the watchdog timer settings for the next application configuration bit to the update register. When the logic array configuration reset (RU_nCONFIG) goes low, the remote system upgrade state machine updates the control register with the contents of the update register and starts system reconfiguration from the new application page.

To ensure successful reconfiguration between the pages, assert the RU_nCONFIG signal for a minimum of 250 ns. This is equivalent to strobing the reconfig input of the ALTREMOTE_UPDATE megafunction high for a minimum of 250 ns.

In the event of an error or reconfiguration trigger condition, the remote system upgrade state machine directs the system to load a factory or application configuration (page zero or page one, based on the mode and error condition) by setting the control register accordingly. Table 9–22 lists the contents of the control register after such an event occurs for all possible error or trigger conditions.

The remote system upgrade status register is updated by the dedicated error monitoring circuitry after an error condition but before the factory configuration is loaded.

Table 9–22. Control Register Contents after an Error or Reconfiguration Trigger Condition (Part 1 of 2)

Reconfiguration Error/Trigger	Control Register Setting Remote Update
nCONFIG reset	All bits are 0
nSTATUS error	All bits are 0
CORE triggered reconfiguration	Update register

Table 9–22. Control Register Contents after an Error or Reconfiguration Trigger Condition (Part 2 of 2)

Reconfiguration Error/Trigger	Control Register Setting Remote Update
CRC error	All bits are 0
wd time out	All bits are 0

Capture operations during factory configuration access the contents of the update register. This feature is used by the user logic to verify that the page address and watchdog timer settings were written correctly. Read operations in application configurations access the contents of the control register. This information is used by the user logic in the application configuration.

User Watchdog Timer

The user watchdog timer prevents a faulty application configuration from stalling the device indefinitely. The system uses the timer to detect functional errors after an application configuration is successfully loaded into the Arria II device.

To allow the remote system upgrade dedicated circuitry to reset the watchdog timer, you must assert the RU_nRSTIMER signal active for a minimum of 250 ns. This is equivalent to strobing the reset_timer input of the ALTREMOTE_UPDATE megafunction high for a minimum of 250 ns.

The user watchdog timer is a counter that counts down from the initial value loaded into the remote system upgrade control register by the factory configuration. The counter is 29 bits wide and has a maximum count value of 2^{29}. When specifying the user watchdog timer value, specify only the most significant 12 bits. The granularity of the timer setting is 2^{17} cycles. The cycle time is based on the frequency of the 10-MHz internal oscillator. Table 9–23 lists the operating range of the 10-MHz internal oscillator.

Table 9–23. 10-MHz Internal Oscillator Specifications

Minimum	Typical	Maximum	Units
4.3	5.3	10	MHz

The user watchdog timer begins counting after the application configuration enters device user mode. This timer must be periodically reloaded or reset by the application configuration before the timer expires by asserting RU_nRSTIMER. If the application configuration does not reload the user watchdog timer before the count expires, a time-out signal is generated by the remote system upgrade dedicated circuitry. The time-out signal tells the remote system upgrade circuitry to set the user watchdog timer status bit (Wd) in the remote system upgrade status register and reconfigures the device by loading the factory configuration.

During the configuration cycle of the device, the user watchdog timer is not enabled. Errors during configuration are detected by the CRC engine. Also, the timer is disabled for factory configurations. Functional errors should not exist in the factory configuration because it is stored and validated during production and is never updated remotely.

The user watchdog timer is disabled in factory configurations and during the configuration cycle of the application configuration. It is enabled after the application configuration enters user mode.

Quartus II Software Support

The Quartus II software provides the flexibility to include the remote system upgrade interface between the Arria II device logic array and the dedicated circuitry, generates configuration files for production, and allows remote programming of the system configuration memory.

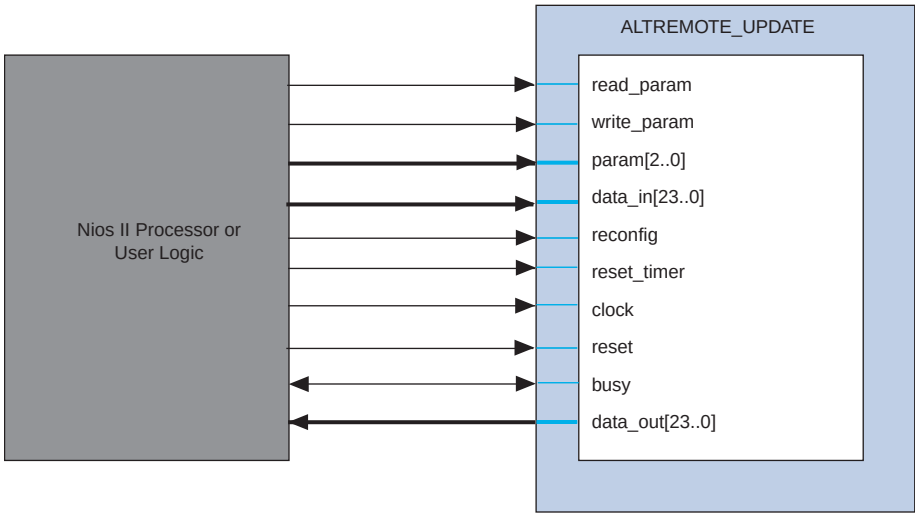
Use the ALTREMOTE_UPDATE megafunction option in the Quartus II software as the interface between the remote system upgrade circuitry and the device logic array interface. Using the megafunction block instead of creating your own logic saves design time and offers more efficient logic synthesis and device implementation.

ALTREMOTE_UPDATE Megafunction

The ALTREMOTE_UPDATE megafunction provides a memory-like interface to the remote system upgrade circuitry and handles the shift register read and write protocol in the Arria II device logic. This implementation is suitable for designs that implement the factory configuration functions using a Nios II processor or user logic in the device.

Figure 9–28 shows the interface signals between the ALTREMOTE_UPDATE megafunction and Nios II processor or user logic.

Figure 9–28. Interface Signals between the ALTREMOTE_UPDATE Megafunction and the Nios II Processor



For more information about the ALTREMOTE_UPDATE megafunction and the description of ports listed in Figure 9–28, refer to the *Remote Update Circuitry (ALTREMOTE_UPDATE) Megafunction User Guide*.

Design Security

This section provides an overview of the design security features and their implementation on Arria II devices using AES. It also covers the new security modes available in Arria II devices.

As Arria II devices continue to play roles in larger and more critical designs in competitive commercial and military environments, it is increasingly important to protect your designs from copying, reverse engineering, and tampering.

Arria II devices address these concerns with both volatile and non-volatile security feature support. Arria II devices have the ability to decrypt configuration bitstreams using the AES algorithm, an industry-standard encryption algorithm that is FIPS-197 certified. Arria II devices have a design security feature which uses a 256-bit security key.

Arria II devices store configuration data in SRAM configuration cells during device operation. Because SRAM memory is volatile, the SRAM cells must be loaded with configuration data each time the device powers up. It is possible to intercept configuration data when it is being transmitted from the memory source (flash memory or a configuration device) to the device. The intercepted configuration data could then be used to configure another device.

When using the Arria II design security feature, the security key is stored in the Arria II device. Depending on the security mode, you can configure the Arria II device using a configuration file that is encrypted with the same key, or for board testing, configured with a normal configuration file.

The design security feature is available when configuring Arria II devices using FPP configuration mode with an external host (such as a MAX II device or microprocessor), or when using AS, fast AS, or PS configuration schemes. The design security feature is also available in remote update mode with AS and fast AS configuration mode.

The design security feature is not available when you are configuring your Arria II device using JTAG-based configuration. For more information, refer to "Supported Configuration Schemes" on page 9–67.

When using a serial configuration scheme such as AS, fast AS, or PS, configuration time is the same whether or not you enable the design security feature. If you use the FPP scheme with the design security or decompression feature, a x4 DCLK is required. This results in a slower configuration time when compared with the configuration time of an Arria II device that has neither the design security nor the decompression feature enabled.

Arria II Security Protection

Arria II device designs are protected from copying, reverse engineering, and tampering using configuration bitstream encryption.

Security Against Copying

The security key is securely stored in the Arria II device and cannot be read out through any interface. In addition, as configuration file read-back is not supported in Arria II devices, your design information cannot be copied.

Security Against Reverse Engineering

Reverse engineering from an encrypted configuration file is very difficult and time consuming because the Arria II configuration file formats are proprietary and the file contains millions of bits which require specific decryption. Reverse engineering the Arria II device is just as difficult because the device is manufactured on the most advanced 40-nm process technology.

Security Against Tampering

After the Tamper Protection bit is set in the key programming file generated by the Quartus II software, the Arria II device can only be configured with configuration files encrypted with the same key. Tampering is prevented using both volatile and non-volatile keys.

AES Decryption Block

The main purpose of the AES decryption block is to decrypt the configuration bitstream prior to entering data decompression or configuration.

Prior to receiving encrypted data, you must enter and store the 256-bit security key in the device. You can choose between a non-volatile security key and a volatile security key with battery backup.

The security key is scrambled prior to storing it in the key storage to make it more difficult for anyone to retrieve the stored key using de-capsulation of the device.

Flexible Security Key Storage

Arria II devices support two types of security key programming—volatile and non-volatile keys. Table 9–24 lists the differences between volatile keys and non-volatile keys.

Table 9–24. Security Key Options

Options	Volatile Key	Non-Volatile Key
Key programmability	Reprogrammable and erasable	One-time programmable
External battery	Required	Not required
Key programming method *(1)*	On-board	On and off board
Design protection	Secure against copying and reverse engineering. Tamper resistant if volatile tamper protection bit is set. *(2)*	Secure against copying and reverse engineering. Tamper resistant if tamper protection bit is set.

Notes to Table 9–24:

(1) Key programming is carried out using the JTAG interface.

(2) Arria II GZ devices do not support this feature.

You can program the non-volatile key to the Arria II device without an external battery. Also, there are no additional requirements of any of the Arria II power supply inputs.

V_{CCBAT} is a dedicated power supply for volatile key storage and not shared with other on-chip power supplies, such as V_{CCIO} or V_{CC}. V_{CCBAT} continuously supplies power to the volatile register regardless of the on-chip supply condition.

For Arria II GX devices, after power up, wait 100 ms (standard POR delay) or 4 ms (fast POR delay) before beginning key programming to ensure that V_{CCBAT} is at full rail. For Arria II GZ devices, after power up, wait 300 ms (PORSEL = 0) or 12 ms (PORSEL = 1) before beginning key programming to ensure that V_{CCBAT} is at full rail.

For more information about how to calculate the key retention time of the battery used for volatile key storage, refer to the *Arria II GX PowerPlay Early Power Estimator*.

For more information about battery specifications, refer to the *Device Datasheet for Arria II Devices* chapter.

For more information about the VCCBAT pin connection recommendations, refer to *Arria II Device Family Pin Connection Guidelines*.

Arria II Design Security Solution

Arria II devices are SRAM-based devices. To provide design security, Arria II devices require a 256-bit security key for configuration bitstream encryption.

To carry out secure configuration, follow these steps (refer to Figure 9–29):

1. Program the security key into the Arria II device.

   Program the user-defined 256-bit AES keys to the Arria II device through the JTAG interface.

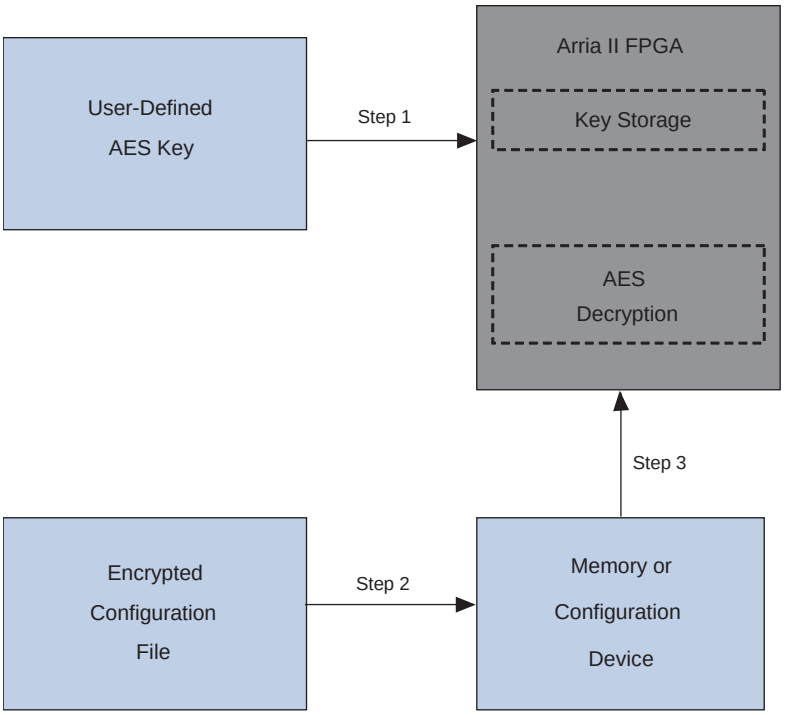
2. Encrypt the configuration file and store it in the external memory.

   Encrypt the configuration file with the same 256-bit keys used to program the Arria II device. Encryption of the configuration file is done using the Quartus II software. The encrypted configuration file is then loaded into the external memory, such as a configuration or flash device.

3. Configure the Arria II device.

At system power-up, the external memory device sends the encrypted configuration file to the Arria II device.

Figure 9–29. Design Security ***(Note 1)***



Note to Figure 9–29:

(1) Step 1, Step 2, and Step 3 correspond to the procedure described in "Design Security" on page 9–62.

Security Modes Available

The following security modes are available on the Arria II device:

Volatile Key

Secure operation with volatile key programmed and required external battery—this mode accepts both encrypted and unencrypted configuration bitstreams. Use the unencrypted configuration bitstream support for board-level testing only.

Non-Volatile Key

Secure operation with one-time-programmable (OTP) security key programmed—this mode accepts both encrypted and unencrypted configuration bitstreams. Use the unencrypted configuration bitstream support for board-level testing only.

Volatile Key with Tamper Protection Bit Set

☞ Arria II GZ devices do not support this feature.

Secure operation in tamper resistant mode with volatile security key programmed—only encrypted configuration bitstreams are allowed to configure the device. Tamper protection disables JTAG configuration with unencrypted configuration bitstream.

☞ Enabling the Tamper Protection bit disables the test mode in Arria II devices. This process is irreversible and prevents Altera from carry-out failure analysis. Contact Altera Technical Support to enable the tamper protection bit.

Non-Volatile Key with Tamper Protection Bit Set

Secure operation in tamper resistant mode with OTP security key programmed—only encrypted configuration bitstreams are allowed to configure the device. Tamper protection disables JTAG configuration with unencrypted configuration bitstream.

☞ Enabling the Tamper Protection bit disables the test mode in Arria II devices. This process is irreversible and prevents Altera from carry-out failure analysis. Contact Altera Technical Support to enable the tamper protection bit.

No Key Operation

Only unencrypted configuration bitstreams are allowed to configure the device.

Table 9–25 lists the different security modes and configuration bitstream supported for each mode.

Table 9–25. Security Modes Supported

Mode *(1)*	Function	Configuration File
Volatile key	Secure	Encrypted
	Board-level testing	Unencrypted
Non-volatile key	Secure	Encrypted
	Board-level testing	Unencrypted
Volatile key with tamper protection bit set *(2)*	Secure (tamper resistant) *(3)*	Encrypted
Non-volatile key with tamper protection bit set	Secure (tamper resistant) *(3)*	Encrypted

Notes to Table 9–25:

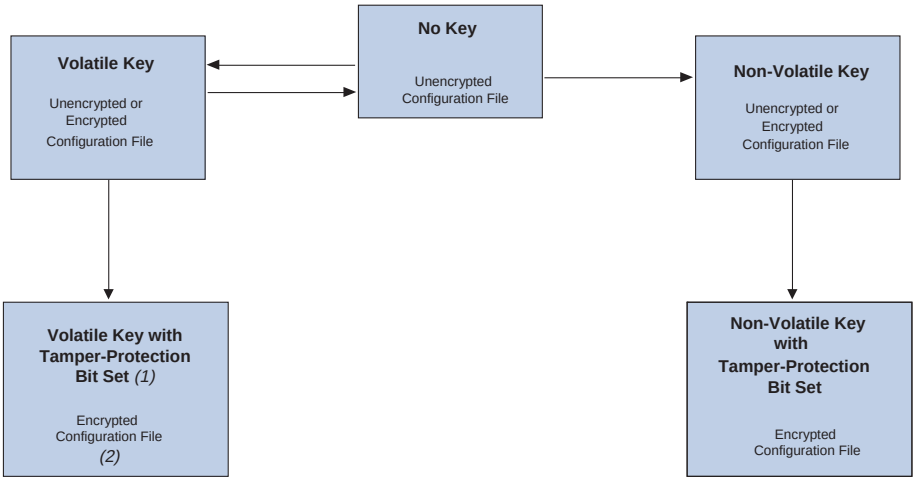
(1) In No key operation, only the unencrypted configuration file is supported.
(2) Arria II GZ devices do not support this feature.
(3) The tamper protection bit setting does not prevent the device from being reconfigured.

Supported Configuration Schemes

The Arria II device supports only selected configuration schemes, depending on the security mode you select when you encrypt the Arria II device.

Figure 9–30 shows the restrictions of each security mode when encrypting Arria II devices.

Figure 9–30. Arria II Security Modes—Sequence and Restrictions



Notes to Figure 9–30:

(1) Arria II GZ devices do not support this feature.
(2) Arria II devices do not accept encrypted configuration files if the volatile key is erased. If the volatile key is earsed, you must use the volatile key without the tamper-protection bit set to reprogram the key.

Table 9–26 lists the configuration modes allowed in each of the security modes.

Table 9–26. Allowed Configuration Modes for Various Security Modes ***(Note 1)***

Security Mode	Configuration File	Allowed Configuration Modes
No key	Unencrypted	All configuration modes that do not engage the design security feature.
Secure with volatile key	Encrypted	■ PS with AES (and/or with decompression) ■ FPP with AES (and/or with decompression) ■ Remote update AS or fast AS with AES (and/or with decompression) ■ AS or fast AS (and/or with decompression)
Board-level testing with volatile key	Unencrypted	All configuration modes that do not engage the design security feature.
Secure with non-volatile key	Encrypted	■ PS with AES (and/or with decompression) ■ FPP with AES (and/or with decompression) ■ Remote update AS or fast AS with AES (and/or with decompression) ■ AS or fast AS (and/or with decompression)
Board-level testing with non-volatile key	Unencrypted	All configuration modes that do not engage the design security feature.
Secure in tamper resistant mode using volatile or non-volatile key with tamper protection set	Encrypted	■ PS with AES (and/or with decompression) ■ FPP with AES (and/or with decompression) ■ Remote update AS or fast AS with AES (and/or with decompression) ■ AS or fast AS (and/or with decompression)

Note to Table 9–26:

(1) There is no impact to the configuration time required when compared with unencrypted configuration modes except when using FPP with AES (and/or decompression), which requires DCLK that is x4 the data rate.

The design security feature is available in all configuration methods except JTAG. Therefore, you can use the design security feature in FPP mode (when using an external controller, such as a MAX II device or a microprocessor and flash memory), or in AS, fast AS, and PS configuration schemes.

Table 9–27 lists the configuration schemes that support the design security feature both for volatile key and non-volatile key programming.

Table 9–27. Design Security Configuration Schemes Availability

Configuration Scheme	Configuration Method	Design Security
FPP	MAX II device or microprocessor and flash memory	✓ *(1)*
AS	Serial configuration device	✓
Fast AS	Serial configuration device	✓
PS	MAX II device or microprocessor and flash memory	✓
	Download cable	✓
JTAG *(2)*	MAX II device or microprocessor and flash memory	—
	Download cable	—

Notes to Table 9–27:

(1) In this mode, the host system must send a DCLK that is x4 the data rate.

(2) JTAG configuration supports only unencrypted configuration file.

You can use the design security feature with other configuration features, such as the compression and remote system upgrade features. When you use compression with the design security feature, the configuration file is first compressed and then encrypted using the Quartus II software. During configuration, the Arria II device first decrypts and then decompresses the configuration file.

Document Revision History

Table 9–28 lists the revision history for this chapter.

Table 9–28. Document Revision History

Date	Version	Changes Made
June 2011	4.1	■ Updated Table 9–9, Table 9–10, Table 9–12, Table 9–18, and Table 9–23. ■ Updated the "Programming Serial Configuration Devices" and "Configuration Data Decompression" sections. ■ Removed references to the "ByteBlaster MV" and "MasterBlaster" cables as they are discontinued. ■ Minor text edits.
December 2010	4.0	■ Updated for the Quartus II software version 10.1 release. ■ Added Arria II GZ devices information. ■ Minor text edits.
July 2010	3.0	Updated for Arria II GX v10.0 release: ■ Updated Table 9–9 and Table 9–17. ■ Updated Figure 9–4, Figure 9–5, Figure 9–13, Figure 9–16, Figure 9–17, Figure 9–21, and Figure 9–30. ■ Updated "AS and Fast AS Configuration (Serial Configuration Devices)" and "Flexible Security Key Storage" sections. ■ Added "Guidelines for Connecting Serial Configuration Device to Arria II Devices on an Active Serial Interface" section. ■ Minor text edits.
November 2009	2.0	Updated for Arria II GX v9.1 release: ■ Updated Table 9–3, Table 9–10, Table 9–11, Table 9–13. ■ Updated Figure 9–2, Figure 9–3, and Figure 9–6. ■ Updated "VCCPD Pins", "JTAG Configuration", "Remote System Upgrade Mode", "Remote System Upgrade State Machine", "User Watchdog Timer" sections. ■ Minor text edits.
June 2009	1.1	■ Updated Table 9–2, Table 9–3, Table 9–9, Table 9–10, Table 9–19, and Table 9–21. ■ Updated Figure 9–6, Figure 9–11, and Figure 9–16. ■ Updated "VCCIO Pins for I/O Banks 3C and 8C", "FPP Configuration Using an External Host", and "Programming Serial Configuration Devices" sections. ■ Removed "Volatite or Non-Volatile Key with JTAG Anti-Tamper Protection Bit Set" section.
February 2009	1.0	Initial release.

11. Configuring Arria GX Devices

AGX52011-1.3

Introduction

Arria™ GX II devices use SRAM cells to store configuration data. Because SRAM memory is volatile, configuration data must be downloaded to Arria GX devices each time the device powers up. Arria GX devices can be configured using one of five configuration schemes: the fast passive parallel (FPP), active serial (AS), passive serial (PS), passive parallel asynchronous (PPA), and Joint Test Action Group (JTAG) configuration schemes. All configuration schemes use either an external controller (for example, a MAX® II device or microprocessor) or a configuration device.

This chapter contains the following sections:

- "Configuration Features" on page 11–4
- "Fast Passive Parallel Configuration" on page 11–13
- "Active Serial Configuration (Serial Configuration Devices)" on page 11–32
- "Passive Serial Configuration" on page 11–44
- "Passive Parallel Asynchronous Configuration" on page 11–71
- "JTAG Configuration" on page 11–82
- "Device Configuration Pins" on page 11–90
- "Conclusion" on page 11–104

Configuration Devices

The Altera® enhanced configuration devices (EPC16, EPC8, and EPC4) support a single-device configuration solution for high-density devices and can be used in the FPP and PS configuration schemes. They are ISP-capable through their JTAG interface. The enhanced configuration devices are divided into two major blocks, the controller and the flash memory.

For information on enhanced configuration devices, refer to the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* and the *Altera Enhanced Configuration Devices* chapters in volume 2 of the *Configuration Handbook*.

The Altera serial configuration devices (EPCS64, EPCS16, and EPCS4) support a single-device configuration solution for Arria GX devices and are used in the AS configuration scheme. Serial configuration devices offer a low cost, low pin count configuration solution.

For information on serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

The EPC2 configuration devices provide configuration support for the PS configuration scheme. The EPC2 device is ISP-capable through its JTAG interface. The EPC2 device can be cascaded to hold large configuration files.

For more information on EPC2 configuration devices, refer to the *Configuration Devices for SRAM-Based LUT Devices Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

Configuration Schemes

The configuration scheme is selected by driving the Arria GX device MSEL pins either high or low, as shown in Table 11–1. The MSEL pins are powered by the V_{CCINT} power supply of the bank they reside in. The MSEL[3..0] pins have 5-kΩ internal pull-down resistors that are always active. During power-on reset (POR) and during reconfiguration, the MSEL pins have to be at LVTTL V_{IL} and V_{IH} levels to be considered a logic low and logic high.

To avoid any problems with detecting an incorrect configuration scheme, hard-wire the MSEL[] pins to V_{CCPD} and GND, without any pull-up or pull-down resistors. Do not drive the MSEL[] pins by a microprocessor or another device.

Table 11–1. Arria GX Configuration Schemes (Part 1 of 2)

Configuration Scheme	MSEL3	MSEL2	MSEL1	MSEL0
Fast passive parallel (FPP)	0	0	0	0
Passive parallel asynchronous (PPA)	0	0	0	1
Passive serial (PS)	0	0	1	0
Remote system upgrade FPP *(1)*	0	1	0	0
Remote system upgrade PPA *(1)*	0	1	0	1
Remote system upgrade PS *(1)*	0	1	1	0
Fast AS (40 MHz) *(2)*	1	0	0	0
Remote system upgrade fast AS (40 MHz) *(2)*	1	0	0	1
FPP with decompression feature enabled *(3)*	1	0	1	1
Remote system upgrade FPP with decompression feature enabled *(1)*, *(3)*	1	1	0	0
AS (20 MHz) *(2)*	1	1	0	1

Table 11–1. Arria GX Configuration Schemes (Part 2 of 2)				
Configuration Scheme	**MSEL3**	**MSEL2**	**MSEL1**	**MSEL0**
Remote system upgrade AS (20 MHz) *(2)*	1	1	1	0
JTAG-based configuration *(5)*	*(4)*	*(4)*	*(4)*	*(4)*

Notes to Table 11–1:

(1) These schemes require that you drive the RUnLU pin to specify either remote update or local update. For more information about remote system upgrades in Arria GX devices, refer to the *Remote System Upgrades With Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*.

(2) Only the EPCS16 and EPCS64 devices support up to a 40 MHz DCLK. Other EPCS devices support up to a 20 MHz DCLK. Refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook* for more information.

(3) These modes are only supported when using a MAX II device or a microprocessor with flash memory for configuration. In these modes, the host system must output a DCLK that is 4× the data rate.

(4) Do not leave the MSEL pins floating. Connect them to V_{CCPD} or ground. These pins support the non-JTAG configuration scheme used in production. If only JTAG configuration is used, you should connect the MSEL pins to ground.

(5) JTAG-based configuration takes precedence over other configuration schemes, which means MSEL pin settings are ignored.

Arria GX devices offer decompression and remote system upgrade features. Arria GX devices can receive a compressed configuration bitstream and decompress this data in real-time, reducing storage requirements and configuration time. You can make real-time system upgrades from remote locations of your Arria GX designs with the remote system upgrade feature.

Table 11–2 shows the uncompressed configuration file sizes for Arria GX devices.

Table 11–2. Arria GX Uncompressed .rbf Sizes *Note (1)*		
Device	**Data Size (Bits)**	**Data Size (MBytes)**
EP1AGX20	9,640,672	1.205
EP1AGX35	9,640,672	1.205
EP1AGX50	16,951,824	2.119
EP1AGX60	16,951,824	2.119
EP1AGX90	25,699,104	3.212

Note to Table 11–2:

(1) **.rbf**: Raw Binary File.

Use the data in Table 11–2 to estimate the file size before design compilation. Different configuration file formats, such as a Hexidecimal (**.hex**) or Tabular Text File (**.ttf**) format, will have different file sizes. However, for any specific version of the Quartus® II software, any design targeted for the same device will have the same uncompressed configuration file size. If you are using compression, the file size can vary after each compilation because the compression ratio is dependent on the design.

This chapter explains the Arria GX device configuration features and describes how to configure Arria GX devices using the supported configuration schemes. This chapter provides configuration pin descriptions and the Arria GX device configuration file formats. In this chapter, the generic term device(s) includes all Arria GX devices.

For more information on setting device configuration options or creating configuration files, refer to the *Software Settings* section in volume 2 of the *Configuration Handbook*.

Configuration Features

Arria GX devices offer configuration data decompression to reduce configuration file storage and remote system upgrades to allow you to remotely update your Arria GX designs. Table 11–3 summarizes which configuration features can be used in each configuration scheme.

Table 11–3. Arria GX Configuration Features

Configuration Scheme	Configuration Method	Decompression	Remote System Upgrade
FPP	MAX II device or a Microprocessor with flash memory	✓ (1)	✓
	Enhanced Configuration Device	✓ (2)	✓
AS	Serial Configuration Device	✓	✓ (3)
PS	MAX II device or a Microprocessor with flash memory	✓	✓
	Enhanced Configuration Device	✓	✓
	Download cable	✓	
PPA	MAX II device or a Microprocessor with flash memory		✓
JTAG	MAX II device or a Microprocessor with flash memory		

Notes to Table 11–3:

(1) In these modes, the host system must send a DCLK that is 4× the data rate.

(2) The enhanced configuration device decompression feature is available, while the Arria GX decompression feature is not available.

(3) Only remote update mode is supported when using the AS configuration scheme. Local update mode is not supported.

Configuration Data Decompression

Arria GX devices support configuration data decompression, which saves configuration memory space and time. This feature allows you to store compressed configuration data in configuration devices or other memory and transmit this compressed bitstream to Arria GX devices. During configuration, Arria GX devices decompress the bitstream in real time and programs its SRAM cells.

Preliminary data indicates that compression typically reduces configuration bitstream size by 35 to 55%.

Arria GX devices support decompression in the FPP (when using a MAX II device/microprocessor + flash), AS, and PS configuration schemes. Decompression is not supported in the PPA configuration scheme nor in JTAG-based configuration.

When using FPP mode, the intelligent host must provide a DCLK that is 4× the data rate. Therefore, the configuration data must be valid for four DCLK cycles.

The decompression feature supported by Arria GX devices is different from the decompression feature in enhanced configuration devices (EPC16, EPC8, and EPC4 devices), although they both use the same compression algorithm. The data decompression feature in the enhanced configuration devices allows them to store compressed data and decompress the bitstream before transmitting it to the target devices. When using Arria GX devices in FPP mode with enhanced configuration devices, the decompression feature is available only in the enhanced configuration device, not the Arria GX device.

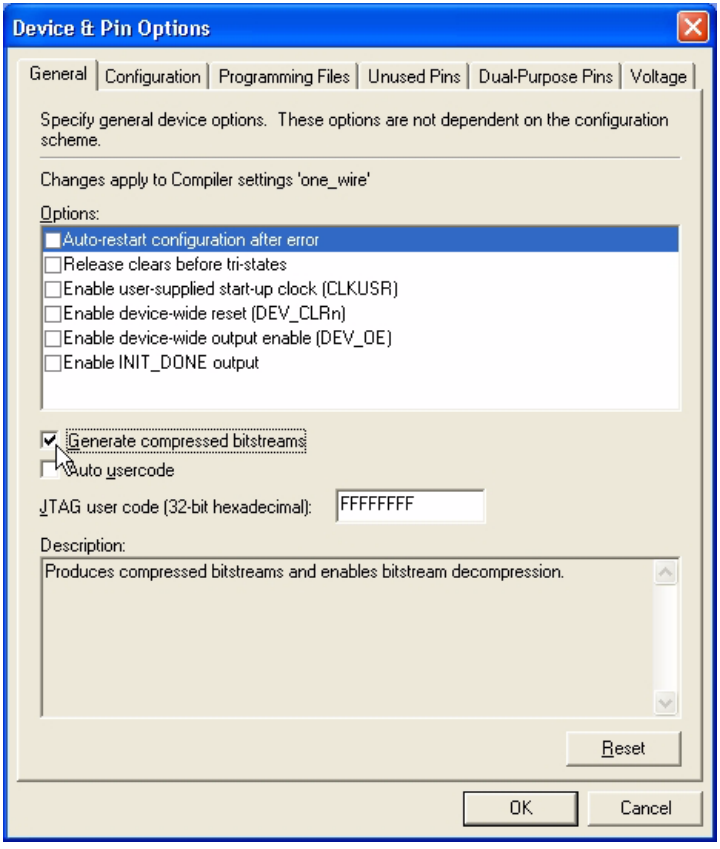
In PS mode, use the Arria GX decompression feature because sending compressed configuration data reduces configuration time. Do not use both the Arria GX device and the enhanced configuration device decompression features simultaneously. The compression algorithm is not intended to be recursive and could expand the configuration file instead of compressing it further.

When you enable compression, the Quartus II software generates configuration files with compressed configuration data. This compressed file reduces the storage requirements in the configuration device or flash memory, and decreases the time needed to transmit the bitstream to the Arria GX device. The time required by an Arria GX device to decompress a configuration file is less than the time needed to transmit the configuration data to the device.

There are two ways to enable compression for Arria GX bitstreams: before design compilation (in the **Compiler Settings** menu) and after design compilation (in the **Convert Programming Files** window).

To enable compression in the project's compiler settings, select **Device** under the **Assignments** menu to bring up the **Settings** window. After selecting your Arria GX device, open the **Device & Pin Options** window, and in the **General** settings tab, enable the check box for **Generate compressed bitstreams** (as shown in Figure 11–1).

Figure 11–1. Enabling Compression for Arria GX Bitstreams in Compiler Settings



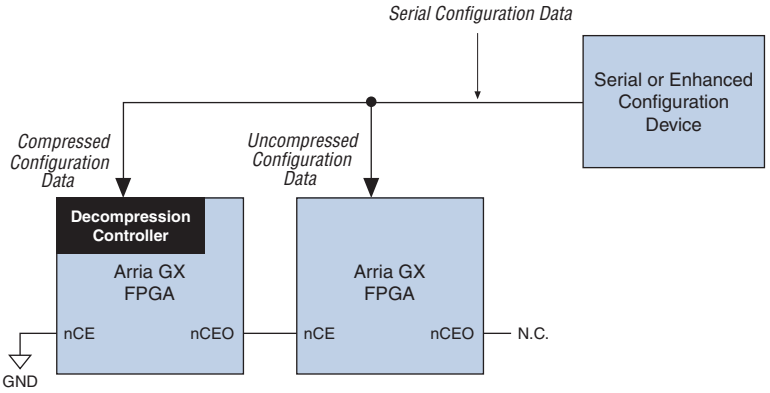
Compression can also be enabled when creating programming files from the **Convert Programming Files** window by following these steps:

1. Click **Conv ert Programming Files** (File menu).
2. Select the programming file type (POF, SRAM HEXOUT, RBF, or TTF).
3. For POF output files, select a configuration device.
4. In the **Input files to convert** box, select **SOF Data**.
5. Select **Add File** and add an Arria GX device SOF(s).
6. Select the name of the file you added to the **SOF Data** area and click **Properties**.
7. Check the **Compression** check box.

When multiple Arria GX devices are cascaded, you can selectively enable the compression feature for each device in the chain if you are using a serial configuration scheme. Figure 11–2 depicts a chain of two Arria GX devices. The first Arria GX device has compression enabled and receives a compressed bitstream from the configuration device. The second Arria GX device has the compression feature disabled and receives uncompressed data.

In a multi-device FPP configuration chain, all Arria GX devices in the chain must either enable of disable the decompression feature. You can not selectively enable the compression feature for each device in the chain because of the DATA and DCLK relationship.

Figure 11–2. Compressed & Uncompressed Configuration Data in the Same Configuration File



You can generate programming files for this setup from the **Convert Programming Files** window (File menu) in the Quartus II software.

Remote System Upgrade

Arria GX devices feature remote and local update.

For more information about this feature, refer to the *Remote System Upgrades with Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*.

Power-On Reset Circuit

The POR circuit keeps the entire system in reset until the power supply voltage levels have stabilized on power-up. Upon power-up, the device does not release nSTATUS until V_{CCINT}, V_{CCPD}, and V_{CCIO} of banks 3, 4, 7, and 8 are above the device's POR trip point. On power down, V_{CCINT} is monitored for brown-out conditions.

The passive serial mode (MSEL[3..0] = 0010) and the Fast passive parallel mode (MSEL[3..0] = 0000) always enable bank 3 to use the lower POR trip point consistent with 1.8- and 1.5-V signaling, regardless of the VCCSEL setting. For all other configuration modes, VCCSEL selects the POR trip point level. Refer to "VCCSEL Pin" on page 11–9 for more details.

In Arria GX devices, the pin-selectable option PORSEL allows you to select between a typical POR time setting of 12 ms or 100 ms. In both cases, you can extend the POR time by using an external component to assert the nSTATUS pin low.

V_{CCPD} Pins

Arria GX devices also offer a new power supply, V_{CCPD}, which must be connected to 3.3-V in order to power the 3.3-V/2.5-V buffer available on the configuration input pins and JTAG pins. V_{CCPD} applies to all the JTAG input pins (TCK, TMS, TDI, and TRST) and the configuration pins when VCCSEL is connected to ground. Refer to Table 11–4 for information on the pins affected by VCCSEL.

V_{CCPD} must ramp-up from 0-V to 3.3-V within 100 ms. If V_{CCPD} is not ramped up within this specified time, your Arria GX device will not configure successfully. If your system does not allow for a V_{CCPD} ramp-up time of 100 ms or less, you must hold nCONFIG low until all power supplies are stable.

VCCSEL Pin

The VCCSEL pin selects the type of input buffer used on configuration input pins and it selects the POR trip point voltage level for V_{CCIO} bank 3 powered by VCCIO3 pins.

The configuration input pins and the PLL_ENA pin (Table 11–4) have a dual buffer design. These pins have a 3.3-V/2.5-V input buffer and a 1.8-V/1.5-V input buffer. The VCCSEL input pin selects which input buffer is used during configuration. The 3.3-V/2.5-V input buffer is powered by V_{CCPD}, while the 1.8-V/1.5-V input buffer is powered by

V_{CCIO}. After configuration, the dual-purpose configuration pins are powered by the V_{CCIO} pins. Table 11–4 shows the pins affected by VCCSEL.

Table 11–4. Pins Affected by the Voltage Level at VCCSEL

Pin	VCCSEL = LOW (connected to GND)	VCCSEL = HIGH (connected to V_{CCPD})
nSTATUS (when used as an input)	3.3/2.5-V input buffer is selected. Input buffer is powered by V_{CCPD}.	1.8/1.5-V input buffer is selected. Input buffer is powered by V_{CCIO} of the I/O bank. These input buffers are 3.3-V tolerant.
nCONFIG		
CONF_DONE (when used as an input)		
DATA[7..0]		
nCE		
DCLK (when used as an input)		
CS		
nWS		
nRS		
nCS		
CLKUSR		
DEV_OE		
DEV_CLRn		
RUnLU		
PLL_ENA		

VCCSEL is sampled during power-up. Therefore, the VCCSEL setting cannot change on-the-fly or during a reconfiguration. The VCCSEL input buffer is powered by V_{CCINT} and has an internal 5-kΩ pull-down resistor that is always active.

VCCSEL must be hardwired to V_{CCPD} or GND.

A logic high selects the 1.8-V/1.5-V input buffer, and a logic low selects the 3.3-V/2.5-V input buffer. VCCSEL should be set to comply with the logic levels driven out of the configuration device or MAX II device or a microprocessor with flash memory.

VCCSEL also sets the POR trip point for I/O bank 3 to ensure that this I/O bank has powered up to the appropriate voltage levels before configuration begins. For passive serial (PS) mode (MSEL[3..0] = 0010) and for Fast passive parallel (FPP) mode (MSEL[3..0] = 0000)

the POR circuitry selects the trip point associated with 1.5/1.8-V signaling. For all other configuration modes defined by MSEL[3..0] settings other than 00X0 (MSEL[1] = X, don't care), VCCSEL=GND selects the higher I/O Bank 3 POR trip point for 2.5V/3.3V signaling and VCCSEL=V_{CCPD} selects the lower I/O Bank 3 POR trip point associated with 1.5V/1.8V signaling.

For all configuration modes with MSEL[3..0] not equal to 00X0 (MSEL[1] = X, don't care), if V_{CCIO} of configuration bank 3 is powered by 1.8-V or 1.5-V and VCCSEL = GND, the voltage supplied to this I/O bank(s) may never reach the POR trip point, which prevents the device from beginning configuration.

The fast passive parallel (FPP) and passive serial (PS) modes always enable bank 3 to use the POR trip point to be consistent with 1.8- and 1.5-V signaling, regardless of the VCCSEL setting.

If the V_{CCIO} of I/O bank 3 is powered by 1.5 or 1.8-V and the configuration signals used require 3.3- or 2.5-V signaling, you should set VCCSEL to V_{CCPD} to enable the 1.8/1.5-V input buffers for configuration. The 1.8-V/1.5-V input buffers are 3.3-V tolerant.

Table 11–5 shows how you should set the VCCSEL, depending on the configuration mode, the voltage level on VCCIO3 pins that power bank 3, and the supported configuration input voltages.

Table 11–5. Supported V_{CCSEL} Setting based on Mode, VCCIO3, and Input Configuration Voltage

Configuration Mode	V_{CCIO} (Bank 3)	Configuration Input Signaling Voltage	V_{CCSEL}
All modes	3.3-V/2.5-V	3.3-V/2.5-V	GND
All modes	1.8-V/1.5-V	3.3-V/2.5-V	V_{CCPD} *(1)*
All modes	1.8-V/1.5-V	1.8-V/1.5-V	V_{CCPD}
-	3.3-V/2.5-V	1.8-V/1.5-V	Not Supported

Note to Table 11–5:

(1) The VCCSEL pin can also be connected to GND for PS (MSEL[3..0]=0010) and FPP (MSEL[3..0]=0000) modes.

The key is to ensure the V_{CCIO} voltage of bank 3 is high enough to trip VCCIO3 POR trip point on power-up. Also, to make sure the configuration device meets the V_{IH} for the configuration input pins based on the selected input buffer.

Table 11–6 shows the configuration mode support for banks 4, 7, and 8.

Table 11–6. Arria GX Configuration Mode Support for Banks 4, 7, & 8

Configuration Mode	Configuration Voltage/V_{CCIO} Support for Banks 4, 7, & 8		
	3.3/3.3	1.8/1.8	3.3/1.8
	VCCSEL = GND	VCCSEL = VCCPD	VCCSEL = GND
Fast passive parallel	Y	Y	Y
Passive parallel asynchronous	Y	Y	Y
Passive serial	Y	Y	Y
Remote system upgrade FPP	Y	Y	Y
Remote system upgrade PPA	Y	Y	Y
Remote system upgrade PS	Y	Y	Y
Fast AS (40 MHz)	Y	Y	Y
Remote system upgrade fast AS (40 MHz)	Y	Y	Y
FPP with decompression	Y	Y	Y
Remote system upgrade FPP with decompression feature enabled	Y	Y	Y
AS (20 MHz)	Y	Y	Y
Remote system upgrade AS (20 MHz)	Y	Y	Y

You must verify the configuration output pins for your chosen configuration modes meet the V_{IH} of the configuration device. Refer to Table 11–22 for a consolidated list of configuration output pins.

The V_{IH} of 3.3 or 2.5 V configuration devices will not be met when the V_{CCIO} of the output configuration pins is 1.8 V or 1.5 V. Level shifters will be required to meet the input high level voltage threshold VIH.

Note that AS mode is only applicable for 3.3-V configuration. If I/O bank 3 is less than 3.3V then level shifters are required on the output pins (DCLK, nCSO, ASDO) from the Arria GX device back to the EPCS device.

The VCCSEL signal does not control TDO or nCEO. During configuration, these pins drive out voltage levels corresponding to the V_{CCIO} supply voltage that powers the I/O bank containing the pin.

For more information on multi-volt support, including information on using TDO and nCEO in multi-volt systems, refer to the *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook*.

Fast Passive Parallel Configuration

Fast passive parallel (FPP) configuration in Arria GX devices is designed to meet the continuously increasing demand for faster configuration times. Arria GX devices are designed with the capability of receiving byte-wide configuration data per clock cycle. Table 11–7 shows the MSEL pin settings when using the FFP configuration scheme.

Table 11–7. Arria GX MSEL Pin Settings for FPP Configuration Schemes

Configuration Scheme	MSEL3	MSEL2	MSEL1	MSEL0
FPP when not using remote system upgrade or decompression feature	0	0	0	0
FPP when using remote system upgrade *(1)*	0	1	0	0
FPP with decompression feature enabled *(2)*	1	0	1	1
FPP when using remote system upgrade and decompression feature *(1)*, *(2)*	1	1	0	0

Notes to Table 11–7:

(1) These schemes require that you drive the RUnLU pin to specify either remote update or local update. For more information about remote system upgrade in Arria GX devices, refer to the *Remote System Upgrades with Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*.

(2) These modes are only supported when using a MAX II device or a microprocessor with flash memory for configuration. In these modes, the host system must output a DCLK that is 4× the data rate.

FPP configuration of Arria GX devices can be performed using an intelligent host, such as a MAX II device, a microprocessor, or an Altera enhanced configuration device.

FPP Configuration Using a MAX II Device as an External Host

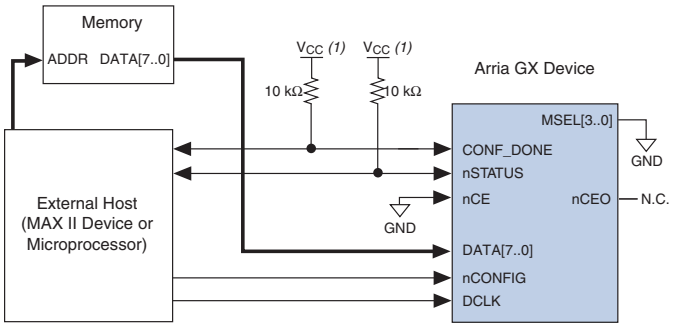
FPP configuration using compression and an external host provides the fastest method to configure Arria GX devices. In the FPP configuration scheme, a MAX II device can be used as an intelligent host that controls the transfer of configuration data from a storage device, such as flash memory, to the target Arria GX device. Configuration data can be stored in RBF, HEX, or TTF format. When using the MAX II devices as an intelligent host, a design that controls the configuration process, such as fetching the data from flash memory and sending it to the device, must be stored in the MAX II device.

If you are using the Arria GX decompression feature, the external host must be able to send a DCLK frequency that is 4× the data rate.

The 4× DCLK signal does not require an additional pin and is sent on the DCLK pin. The maximum DCLK frequency is 100 MHz, which results in a maximum data rate of 200 Mbps. If you are not using the Arria GX decompression feature, the data rate is 8× the DCLK frequency.

Figure 11–3 shows the configuration interface connections between the Arria GX device and a MAX II device for single device configuration.

Figure 11–3. Single Device FPP Configuration Using an External Host



Note to Figure 11–3:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for the device. V_{CC} should be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

Upon power-up, the Arria GX devices go through a power-on reset (POR). The POR delay is dependent on the PORSEL pin setting: when PORSEL is driven low, the POR time is approximately 100 ms; when PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device resets, holds nSTATUS low, and tri-states all user I/O pins. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins have weak pull-up resistors, which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *DC & Switching Characteristics* chapter in volume 1 of the *Arria GX Device Handbook*.

The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in the reset stage. To initiate configuration, the MAX II device must drive the nCONFIG pin from low to high.

V_{CCINT}, V_{CCIO}, and V_{CCPD} of the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. Once nSTATUS is released, the device is ready to receive configuration data and the configuration stage begins. When nSTATUS is pulled high, the MAX II device places the configuration data, one byte at a time, on the DATA[7..0] pins.

Arria GX devices receive configuration data on the DATA[7..0] pins and the clock is received on the DCLK pin. Data is latched into the device on the rising edge of DCLK. If you are using the Arria GX decompression feature, configuration data is latched on the rising edge of every fourth DCLK cycle. After the configuration data is latched in, it is processed during the following three DCLK cycles.

Data is continuously clocked into the target device until CONF_DONE goes high. The CONF_DONE pin goes high one byte early in parallel configuration (FPP and PPA) modes. The last byte is required for serial configuration (AS and PS) modes. After the device has received the next to last byte of the configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by an external 10-kΩ pull-up resistor. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

In Arria GX devices, the initialization clock source is either the internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Arria GX device provides itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device. Driving DCLK to the device after configuration is complete does not affect device operation.

You can also synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. The **Enable user-supplied start-up clock** (**CLKUSR**) option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR does not affect the configuration process. The CONF_DONE pin goes high one byte early in parallel configuration (FPP and PPA) modes. The last byte is required for serial configuration (AS and PS) modes. After the CONF_DONE pin transitions high, CLKUSR is enabled after the time specified as t_{CD2CU}. After this time period elapses, Arria GX devices require 299 clock cycles to initialize properly and enter user mode. Arria GX devices support a CLKUSR f_{MAX} of 100 MHz.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. This **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used, it is high because of an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. The MAX II device must be able to detect this low-to-high transition, which signals the device has entered user mode. When initialization is complete, the device enters user mode. In user-mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

To ensure DCLK and DATA[7..0] are not left floating at the end of configuration, the MAX II device must drive them either high or low, whichever is convenient on your board. The DATA[7..0] pins are available as user I/O pins after configuration. When you select the FPP scheme in the Quartus II software, as a default, these I/O pins are tri-stated in user mode. To change this default option in the Quartus II software, select the **Pins** tab of the **Device & Pin Options** dialog box.

The configuration clock (DCLK) speed must be below the specified frequency to ensure correct configuration. No maximum DCLK period exists, which means you can pause configuration by halting DCLK for an indefinite amount of time.

If you are using the Arria GX decompression feature and need to stop DCLK, it can only be stopped three clock cycles after the last data byte was latched into the Arria GX device.

By stopping DCLK, the configuration circuit allows enough clock cycles to process the last byte of latched configuration data. When the clock restarts, the MAX II device must provide data on the DATA[7..0] pins prior to sending the first DCLK rising edge.

If an error occurs during configuration, the device drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the MAX II device that there is an error. If the **Auto-restart configuration after error** option (available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box) is turned on, the device releases nSTATUS after a reset time-out period (maximum of 100 µs). After nSTATUS is released and pulled high by a pull-up resistor, the MAX II device can try to reconfigure the target device

without needing to pulse nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

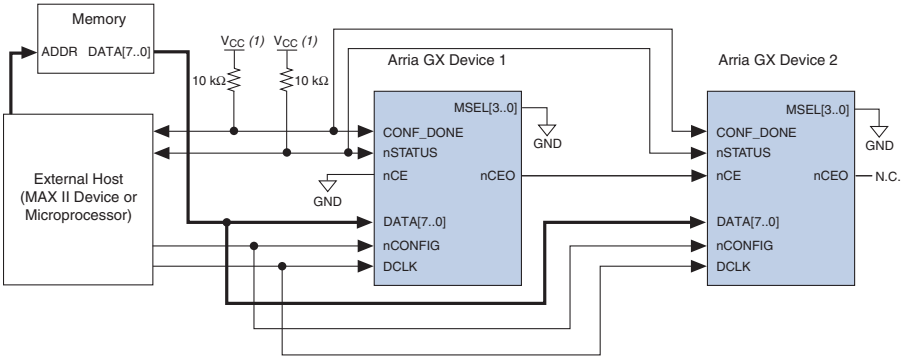
The MAX II device can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The CONF_DONE pin must be monitored by the MAX II device to detect errors and determine when programming completes. If all configuration data is sent, but the CONF_DONE or INIT_DONE signals have not gone high, the MAX II device will reconfigure the target device.

If the optional CLKUSR pin is used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure CLKUSR continues toggling during the time nSTATUS is low (maximum of 100 µs).

When the device is in user-mode, initiating a reconfiguration is done by transitioning the nCONFIG pin low-to-high. The nCONFIG pin should be low for at least 2 µs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Once nCONFIG returns to a logic high level and nSTATUS is released by the device, reconfiguration begins.

Figure 11–4 shows how to configure multiple devices using a MAX II device. This circuit is similar to the FPP configuration circuit for a single device, except the Arria GX devices are cascaded for multi-device configuration.

Figure 11–4. Multi-Device FPP Configuration Using an External Host



Note to Figure 11–4:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain. V_{CC} should be high enough to meet the V_{IH} specification of the I/O standard on the device and the external host.

In multi-device FPP configuration, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration within one clock cycle; therefore, the transfer of data destinations is transparent to the MAX II device. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. The configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

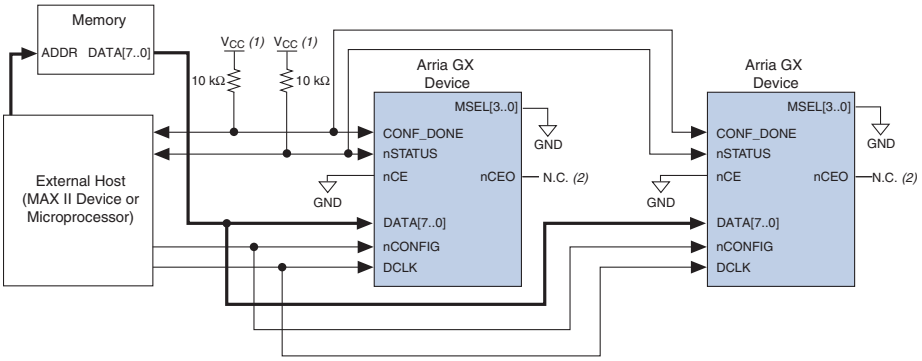
All nSTATUS and CONF_DONE pins are tied together and if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

If the **Auto-restart configuration after error** option is turned on, the devices release their nSTATUS pins after a reset time-out period (maximum of 100 µs). After all nSTATUS pins are released and pulled high, the MAX II device can try to reconfigure the chain without pulsing nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

In a multi-device FPP configuration chain, all Arria GX devices in the chain must either enable or disable the decompression feature. You can not selectively enable the decompression feature for each device in the chain because of the DATA and DCLK relationship.

If a system has multiple devices that contain the same configuration data, tie all device nCE inputs to GND, and leave nCEO pins floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. Configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices start and complete configuration at the same time. Figure 11–5 shows multi-device FPP configuration when both Arria GX devices are receiving the same configuration data.

Figure 11–5. Multiple-Device FPP Configuration Using an External Host When Both Devices Receive the Same Data



Notes to Figure 11–5:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain. V_{CC} should be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

(2) The nCEO pins of both Arria GX devices are left unconnected when configuring the same configuration data into multiple devices.

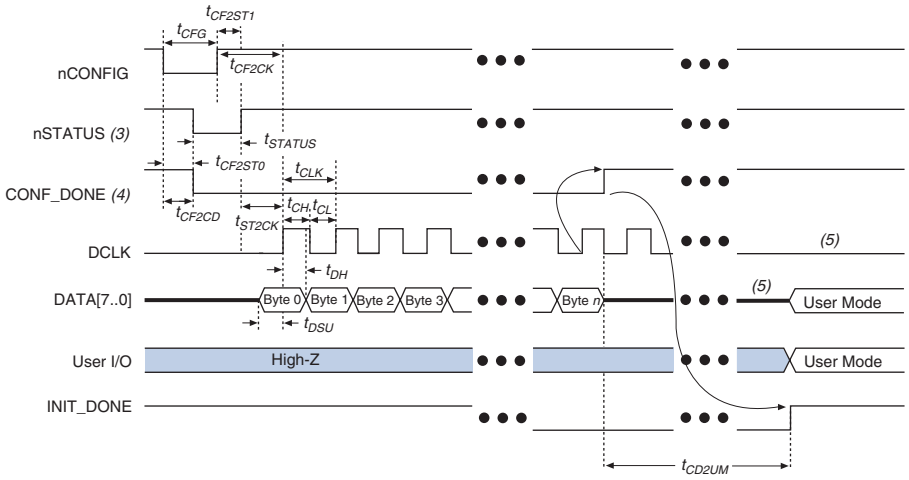
You can use a single configuration chain to configure Arria GX devices with other Altera devices that support FPP configuration, such as Stratix® devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, tie all of the device CONF_DONE and nSTATUS pins together.

For more information about configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook.*

FPP Configuration Timing

Figure 11–6 shows the timing waveform for FPP configuration when using a MAX II device as an external host. This waveform shows the timing when the decompression feature is not enabled.

Figure 11–6. FPP Configuration Timing Waveform *Notes (1), (2)*



Notes to Figure 11–6:

(1) This timing waveform should be used when the decompression feature is not used.
(2) The beginning of this waveform shows the device in user-mode. In user-mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.
(3) Upon power-up, the Arria GX device holds nSTATUS low for the time of the POR delay.
(4) Upon power-up, before and during configuration, CONF_DONE is low.
(5) DCLK should not be left floating after configuration. It should be driven high or low, whichever is more convenient.
(6) DATA[7..0] are available as user I/O pins after configuration and the state of these pins depends on the dual-purpose pin settings.

Table 11–8 defines the timing parameters for Arria GX devices for FPP configuration when the decompression feature is not enabled.

Table 11–8. FPP Timing Parameters for Arria GX Devices (Part 1 of 2) *Notes (1), (2)*

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		800	ns
t_{CFG}	nCONFIG low pulse width	2		µs
t_{STATUS}	nSTATUS low pulse width	10	100 *(3)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high		100 *(3)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	100		µs
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2		µs
t_{DSU}	Data setup time before rising edge on DCLK	5		ns

Table 11–8. FPP Timing Parameters for Arria GX Devices (Part 2 of 2) *Notes (1), (2)*

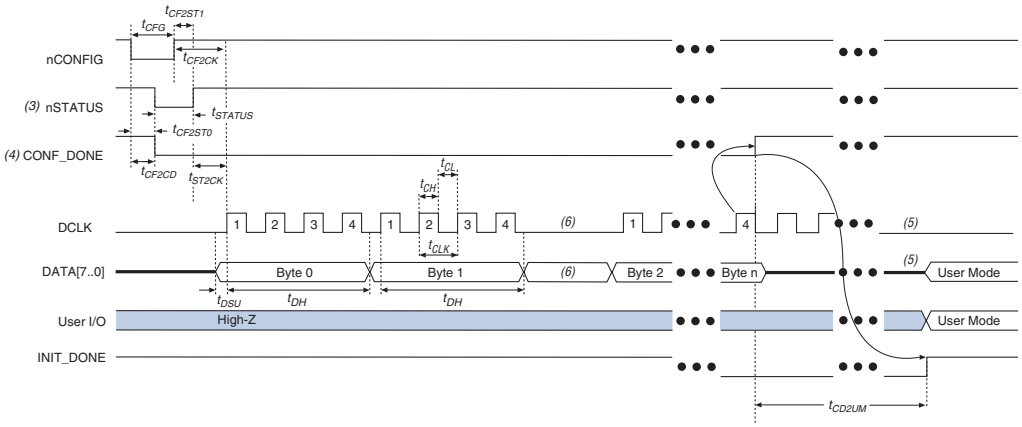
Symbol	Parameter	Min	Max	Units
t_{DH}	Data hold time after rising edge on DCLK	0		ns
t_{CH}	DCLK high time	4		ns
t_{CL}	DCLK low time	4		ns
t_{CLK}	DCLK period	10		ns
f_{MAX}	DCLK frequency		100	MHz
t_R	Input rise time		40	ns
t_F	Input fall time		40	ns
t_{CD2UM}	CONF_DONE high to user mode *(4)*	20	100	µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period		
t_{CD2UMC}	CONF_DONE high to user mode with **CLKUSR** option on	t_{CD2CU} +(299 × CLKUSR period)		

Notes to Table 11–8:

(1) This information is preliminary.
(2) These timing parameters should be used when the decompression feature is not used.
(3) This value is obtainable if you do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.
(4) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device.

Figure 11–7 shows the timing waveform for FPP configuration when using a MAX II device as an external host. This waveform shows the timing when the decompression feature is enabled.

Figure 11–7. FPP Configuration Timing Waveform With Decompression Feature Enabled *Notes (1), (2)*



Notes to Figure 11–7:

(1) This timing waveform should be used when the decompression feature is used.

(2) The beginning of this waveform shows the device in user-mode. In user-mode, nCONFIG, nSTATUS and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.

(3) Upon power-up, the Arria GX device holds nSTATUS low for the time of the POR delay.

(4) Upon power-up, before and during configuration, CONF_DONE is low.

(5) DCLK should not be left floating after configuration. It should be driven high or low, whichever is more convenient.

(6) DATA[7..0] are available as user I/O pins after configuration and the state of these pins depends on the dual-purpose pin settings. If needed, DCLK can be paused by holding it low. When DCLK restarts, the external host must provide data on the DATA[7..0] pins prior to sending the first DCLK rising edge.

Table 11–9 defines the timing parameters for Arria GX devices for FPP configuration when the decompression feature is enabled.

Table 11–9. FPP Timing Parameters for Arria GX Devices With Decompression Feature Enabled *Notes (1), (2)*

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		800	ns
t_{CFG}	nCONFIG low pulse width	2		µs
t_{STATUS}	nSTATUS low pulse width	10	100 *(3)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high		100 *(3)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	100		µs
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2		µs
t_{DSU}	Data setup time before rising edge on DCLK	5		ns
t_{DH}	Data hold time after rising edge on DCLK	30		ns
t_{CH}	DCLK high time	4		ns
t_{CL}	DCLK low time	4		ns
t_{CLK}	DCLK period	10		ns
f_{MAX}	DCLK frequency		100	MHz
t_{DATA}	Data rate		200	Mbps
t_R	Input rise time		40	ns
t_F	Input fall time		40	ns
t_{CD2UM}	CONF_DONE high to user mode *(4)*	20	100	µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period		
t_{CD2UMC}	CONF_DONE high to user mode with **CLKUSR** option on	t_{CD2CU} + (299 × CLKUSR period)		

Notes to Table 11–9:

(1) This information is preliminary.
(2) These timing parameters should be used when the decompression feature is used.
(3) This value is obtainable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.
(4) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device.

Device configuration options and how to create configuration files are discussed further in the *Software Settings* section in the *Configuration Handbook*.

FPP Configuration Using a Microprocessor

In the FPP configuration scheme, a microprocessor can control the transfer of configuration data from a storage device, such as flash memory, to the target Arria GX device.

All information in "FPP Configuration Using a MAX II Device as an External Host" on page 11–13 is also applicable when using a microprocessor as an external host. Refer to that section for all configuration and timing information.

FPP Configuration Using an Enhanced Configuration Device

In the FPP configuration scheme, an enhanced configuration device sends a byte of configuration data every DCLK cycle to the Arria GX device. Configuration data is stored in the configuration device.

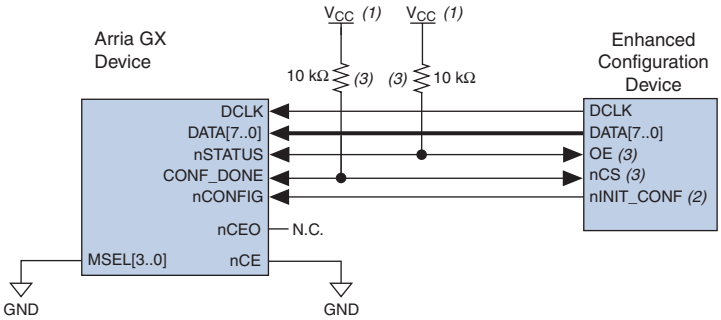
When configuring your Arria GX device using FPP mode and an enhanced configuration device, the enhanced configuration device decompression feature is available while the Arria GX decompression feature is not.

Figure 11–8 shows the configuration interface connections between a Arria GX device and the enhanced configuration device for single device configuration.

The figures in this chapter only show the configuration-related pins and the configuration pin connections between the configuration device and the device.

For more information on the enhanced configuration device and flash interface pins, such as PGM[2..0], EXCLK, PORSEL, A[20..0], and DQ[15..0], refer to the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* in the *Configuration Handbook*.

Figure 11–8. Single Device FPP Configuration Using an Enhanced Configuration Device



Notes to Figure 11–8:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active. This means an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor. If reconfiguration is required, a resistor is necessary.

(3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

The value of the internal pull-up resistors on the enhanced configuration devices can be found in the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* in the *Configuration Handbook*.

When using enhanced configuration devices, you can connect the device's nCONFIG pin to nINIT_CONF pin of the enhanced configuration device, which allows the INIT_CONF JTAG instruction to initiate device configuration. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor. An internal pull-up resistor on the nINIT_CONF pin is always active in the enhanced configuration devices, which means an external pull-up resistor should not be used if nCONFIG is tied to nINIT_CONF.

Upon power-up, the Arria GX device goes through a POR. The POR delay is dependent on the PORSEL pin setting: when PORSEL is driven low, the POR time is approximately 100 ms; when PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device will reset, hold

nSTATUS low, and tri-state all user I/O pins. The configuration device also goes through a POR delay to allow the power supply to stabilize. The POR time for enhanced configuration devices can be set to either 100 ms or 2 ms, depending on its PORSEL pin setting. If the PORSEL pin is connected to GND, the POR delay is 100 ms. If the PORSEL pin is connected to V_{CC}, the POR delay is 2 ms. During this time, the configuration device drives its OE pin low. This low signal delays configuration because the OE pin is connected to the target device's nSTATUS pin.

When selecting a POR time, you need to ensure that the device completes power-up before the enhanced configuration device exits POR. Altera recommends that you use a 12-ms POR time for the Arria GX device, and use a 100-ms POR time for the enhanced configuration device.

When both devices complete POR, they release their open-drain OE or nSTATUS pin, which is then pulled high by a pull-up resistor. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors, which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *Arria GX Device Handbook*.

When the power supplies have reached the appropriate operating voltages, the target device senses the low-to-high transition on nCONFIG and initiates the configuration cycle. The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. The beginning of configuration can be delayed by holding the nCONFIG or nSTATUS pin low.

V_{CCINT}, V_{CCIO}, and V_{CCPD} of the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the nSTATUS pin, which is pulled high by a pull-up resistor. Enhanced configuration devices have an optional internal pull-up resistor on the OE pin. This option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If this internal pull-up resistor is not used, an external 10-kΩ pull-up resistor on the OE-nSTATUS line is required. Once nSTATUS is released, the device is ready to receive configuration data and the configuration stage begins.

When nSTATUS is pulled high, the configuration device's OE pin also goes high and the configuration device clocks data out to the device using the Arria GX device's internal oscillator. The Arria GX devices receive configuration data on the DATA[7..0] pins and the clock is received on the DCLK pin. A byte of data is latched into the device on each rising edge of DCLK.

After the device has received all configuration data successfully, it releases the open-drain CONF_DONE pin which is pulled high by a pull-up resistor. Because CONF_DONE is tied to the configuration device's nCS pin, the configuration device is disabled when CONF_DONE goes high. Enhanced configuration devices have an optional internal pull-up resistor on the nCS pin. This option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If this internal pull-up resistor is not used, an external 10-kΩ pull-up resistor on the nCS-CONF_DONE line is required. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin.

In Arria GX devices, the initialization clock source is either the internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Arria GX device provides itself with enough clock cycles for proper initialization. You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. The **Enable user-supplied start-up clock** (**CLKUSR**) option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR will not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, CLKUSR will be enabled after the time specified as t_{CD2CU}. After this time period elapses, Arria GX devices require 299 clock cycles to initialize properly and enter user mode. Arria GX devices support a CLKUSR f_{MAX} of 100 MHz.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used, it will be high due to an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low. When initialization is complete, the INIT_DONE pin will be released and pulled high. In user-mode, the user

I/O pins will no longer have weak pull-up resistors and will function as assigned in your design. The enhanced configuration device will drive DCLK low and DATA[7..0] high at the end of configuration.

If an error occurs during configuration, the device drives its nSTATUS pin low, resetting itself internally. Because the nSTATUS pin is tied to OE, the configuration device will also be reset. If the **Auto-restart configuration after error** option (available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box) is turned on, the device will automatically initiate reconfiguration if an error occurs. The Arria GX device releases its nSTATUS pin after a reset time-out period (maximum of 100 µs). When the nSTATUS pin is released and pulled high by a pull-up resistor, the configuration device reconfigures the chain. If this option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 2 µs to restart configuration. The external system can pulse nCONFIG if nCONFIG is under system control rather than tied to V_{CC}.

In addition, if the configuration device sends all of its data and then detects that CONF_DONE has not gone high, it recognizes that the device has not configured successfully. Enhanced configuration devices wait for 64 DCLK cycles after the last configuration bit was sent for CONF_DONE to reach a high state. In this case, the configuration device pulls its OE pin low, which in turn drives the target device's nSTATUS pin low. If the Auto-restart configuration after error option is set in the software, the target device resets and then releases its nSTATUS pin after a reset time-out period (maximum of 100 µs). When nSTATUS returns to a logic high level, the configuration device will try to reconfigure the device.

When CONF_DONE is sensed low after configuration, the configuration device recognizes that the target device has not configured successfully. Therefore, your system should not pull CONF_DONE low to delay initialization. Instead, you should use the CLKUSR option to synchronize the initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain will initialize together if their CONF_DONE pins are tied together.

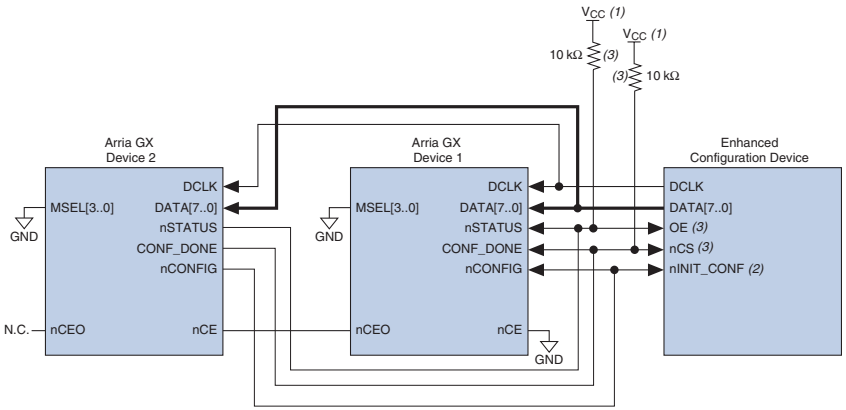
If the optional CLKUSR pin is used and nCONFIG is pulled low to restart configuration during device initialization, ensure CLKUSR continues toggling during the time nSTATUS is low (maximum of 100 µs).

When the device is in user-mode, a reconfiguration can be initiated by pulling the nCONFIG pin low. The nCONFIG pin should be low for at least 2 µs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Because CONF_DONE is

pulled low, this activates the configuration device because it sees its nCS pin drive low. Once nCONFIG returns to a logic high level and nSTATUS is released by the device, reconfiguration begins.

Figure 11–9 shows how to configure multiple Arria GX devices with an enhanced configuration device. This circuit is similar to the configuration device circuit for a single device, except the Arria GX devices are cascaded for multi-device configuration.

Figure 11–9. Multi-Device FPP Configuration Using an Enhanced Configuration Device



Notes to Figure 11–9:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active. This means an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-up resistors on configuration device** option when generating programming files.

Enhanced configuration devices cannot be cascaded.

When performing multi-device configuration, you must generate the configuration device's POF from each project's SOF. You can combine multiple SOFs using the **Convert Programming Files** window in the Quartus II software.

For more information on how to create configuration files for multi-device configuration chains, refer to the *Software Settings* section in volume 2 of the *Configuration Handbook*.

In multi-device FPP configuration, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. Pay special attention to the configuration signals because they may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device.

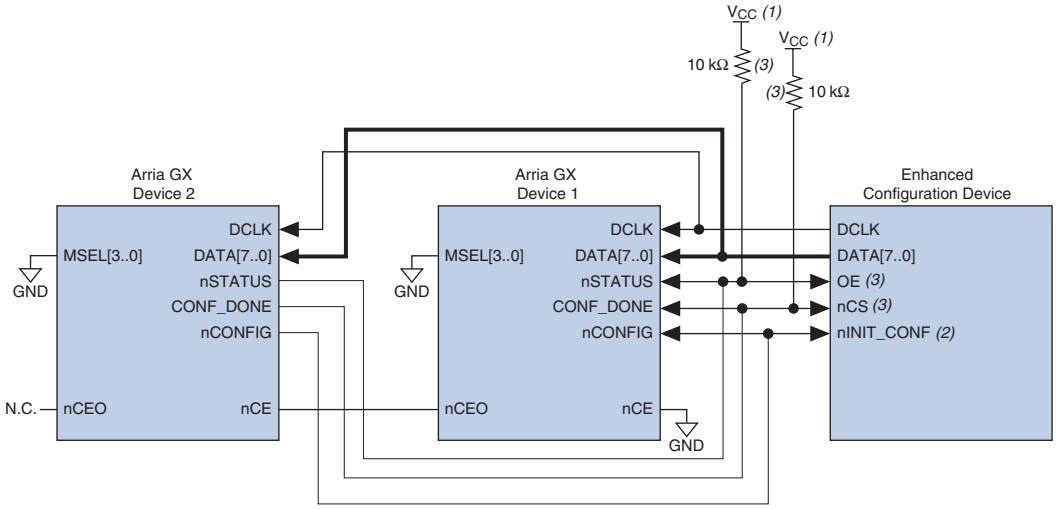
When configuring multiple devices, configuration does not begin until all devices release their OE or nSTATUS pins. Similarly, because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Because all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This low signal drives the OE pin low on the enhanced configuration device and drives nSTATUS low on all devices, which causes them to enter a reset state. This behavior is similar to a single device detecting an error.

If the **Auto-restart configuration after error** option is turned on, the devices will automatically initiate reconfiguration if an error occurs. The devices will release their nSTATUS pins after a reset time-out period (maximum of 100 µs). When all the nSTATUS pins are released and pulled high, the configuration device tries to reconfigure the chain. If the **Auto-restart configuration after error** option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 2 µs to restart configuration. The external system can pulse nCONFIG if nCONFIG is under system control rather than tied to V_{CC}.

Your system may have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. Configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete configuration at the same time. Figure 11–10 shows multi-device FPP configuration when both Arria GX devices are receiving the same configuration data.

Figure 11–10. Multiple-Device FPP Configuration Using an Enhanced Configuration Device When Both Devices Receive the Same Data



Notes to Figure 11–10:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active. This means an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

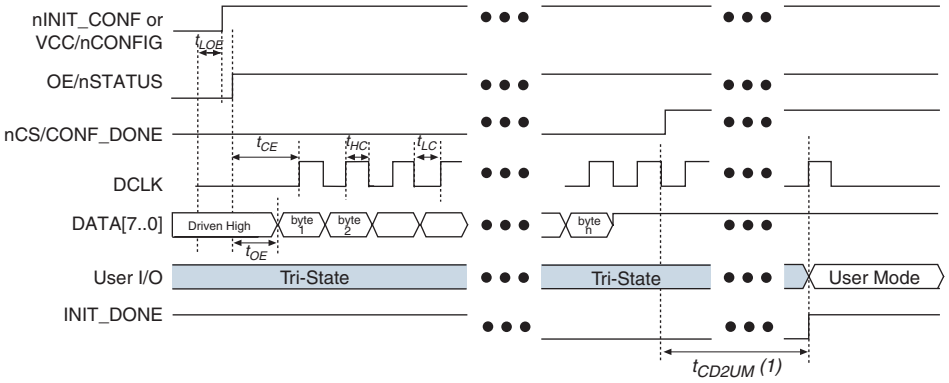
(4) The nCEO pins of both devices are left unconnected when configuring the same configuration data into multiple devices.

You can use a single enhanced configuration chain to configure multiple Arria GX devices with other Altera devices that support FPP configuration, such as Arria GX devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information about configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook*.

Figure 11–11 shows the timing waveform for the FPP configuration scheme using an enhanced configuration device.

Figure 11–11. Arria GX FPP Configuration Using an Enhanced Configuration Device Timing Waveform



Note to Figure 11–11:
(1) The initialization clock can come from the Arria GX device's internal oscillator or the CLKUSR pin.

For timing information, refer to the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* in volume 2 of the *Configuration Handbook*.

Device configuration options and how to create configuration files are discussed further in the *Software Settings* section of the *Configuration Handbook*.

Active Serial Configuration (Serial Configuration Devices)

In the AS configuration scheme, Arria GX devices are configured using a serial configuration device. These configuration devices are low-cost devices with non-volatile memory that feature a simple four-pin interface and a small form factor. These features make serial configuration devices an ideal low-cost configuration solution.

For more information on serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook*.

Serial configuration devices provide a serial interface to access configuration data. During device configuration, Arria GX devices read configuration data via the serial interface, decompresses data if necessary, and configures their SRAM cells. This scheme is referred to as the AS

configuration scheme because the device controls the configuration interface. This scheme contrasts with the PS configuration scheme, where the configuration device controls the interface.

The Arria GX decompression feature is fully available when configuring your Arria GX device using AS mode.

Table 11–10 shows the MSEL pin settings when using the AS configuration scheme.

Table 11–10. Arria GX MSEL Pin Settings for AS Configuration Schemes

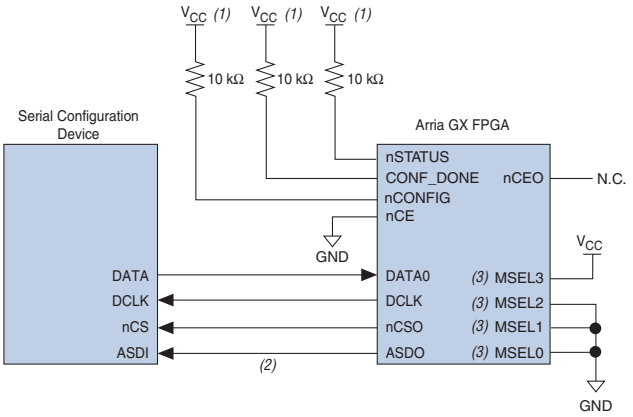
Configuration Scheme	MSEL3	MSEL2	MSEL1	MSEL0
Fast AS (40 MHz) *(1)*	1	0	0	0
Remote system upgrade fast AS (40 MHz) *(1)*	1	0	0	1
AS (20 MHz) *(1)*	1	1	0	1
Remote system upgrade AS (20 MHz) *(1)*	1	1	1	0

Note to Table 11–10:

(1) Only the EPCS16 and EPCS64 devices support a DCLK up to 40 MHz clock; other EPCS devices support a DCLK up to 20 MHz. Refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook* for more information.

Serial configuration devices have a four-pin interface: serial clock input (DCLK), serial data output (DATA), AS data input (ASDI), and an active-low chip select (nCS). This four-pin interface connects to Arria GX device pins, as shown in Figure 11–12.

Figure 11–12. Single Device AS Configuration



Notes to Figure 11–12:

(1) Connect the pull-up resistors to a 3.3-V supply.

(2) Arria GX devices use the ASDO to ASDI path to control the configuration device.

(3) If using an EPCS4 device, MSEL[3..0] should be set to **1101**. Refer to Table 11–10 for more details.

Upon power-up, Arria GX devices go through a POR. The POR delay is dependent on the PORSEL pin setting. When PORSEL is driven low, the POR time is approximately 100 ms. If PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device will reset, hold nSTATUS and CONF_DONE low, and tri-state all user I/O pins. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *DC & Switching Characteristics* chapter in volume 1 of the *Arria GX Device Handbook*.

The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. After POR, the Arria GX devices release nSTATUS, which is pulled high by an external 10-kΩ pull-up resistor, and enters configuration mode.

To begin configuration, power the V_{CCINT}, V_{CCIO}, and V_{CCPD} voltages (for the banks where the configuration and JTAG pins reside) to the appropriate voltage levels.

The serial clock (DCLK) generated by Arria GX devices controls the entire configuration cycle and provides the timing for the serial interface. Arria GX devices use an internal oscillator to generate DCLK. Using the MSEL[] pins, you can select to use either a 40- or 20-MHz oscillator.

Only the EPCS16 and EPCS64 devices support a DCLK up to 40-MHz clock; other EPCS devices support a DCLK up to 20-MHz.

Refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook* for more information.

The EPCS4 device only supports the smallest Arria GX (EP2S15) device, which is when the SOF compression is enabled. Because of its insufficient memory capacity, the EPCS1 device does not support any Arria GX devices.

Table 11–11 shows the active serial DCLK output frequencies.

Table 11–11. Active Serial DCLK Output Frequency *Note (1)*

Oscillator	Minimum	Typical	Maximum	Units
40 MHz *(2)*	20	26	40	MHz
20 MHz	10	13	20	MHz

Notes to Table 11–11:

(1) These values are preliminary.

(2) Only the EPCS16 and EPCS64 devices support a DCLK up to 40-MHz clock; other EPCS devices support a DCLK up to 20-MHz. Refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook* for more information.

In both AS and fast AS configuration schemes, the serial configuration device latches input and control signals on the rising edge of DCLK and drives out configuration data on the falling edge. Arria GX devices drive out control signals on the falling edge of DCLK and latch configuration data on the falling edge of DCLK.

In configuration mode, Arria GX devices enable the serial configuration device by driving the nCSO output pin low, which connects to the chip select (nCS) pin of the configuration device. Arria GX devices use the serial clock (DCLK) and serial data output (ASDO) pins to send operation commands and/or read address signals to the serial configuration device. The configuration device provides data on its serial data output (DATA) pin, which connects to the DATA0 input of the Arria GX devices.

After all configuration bits are received by the Arria GX device, it releases the open-drain CONF_DONE pin, which is pulled high by an external 10-kΩ resistor. Initialization begins only after the CONF_DONE signal reaches a logic high level. All AS configuration pins (DATA0, DCLK, nCSO, and ASDO) have weak internal pull-up resistors that are always active. After configuration, these pins are set as input tri-stated and are driven high by the weak internal pull-up resistors. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

In Arria GX devices, the initialization clock source is either the 10-MHz (typical) internal oscillator (separate from the active serial internal oscillator) or the optional **CLKUSR** pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Arria GX device provides itself with enough clock cycles for proper initialization. You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. The **Enable user-supplied start-up clock** (**CLKUSR**) option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. When you **Enable** the **user supplied start-up clock** option, the CLKUSR pin is the initialization clock source. Supplying a clock on CLKUSR will not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, CLKUSR is enabled after 600 ns. After this time period elapses, Arria GX devices require 299 clock cycles to initialize properly and enter user mode. Arria GX devices support a CLKUSR f_{MAX} of 100 MHz.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used, it will be high due to an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. This low-to-high transition signals that the device has entered user mode. When initialization is complete, the device enters user mode. In user mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

If an error occurs during configuration, Arria GX devices assert the nSTATUS signal low, indicating a data frame error, and the CONF_DONE signal stays low. If the **Auto-restart configuration after error** option (available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box) is turned on, the Arria GX device resets the configuration device by pulsing nCSO, releases nSTATUS after a reset

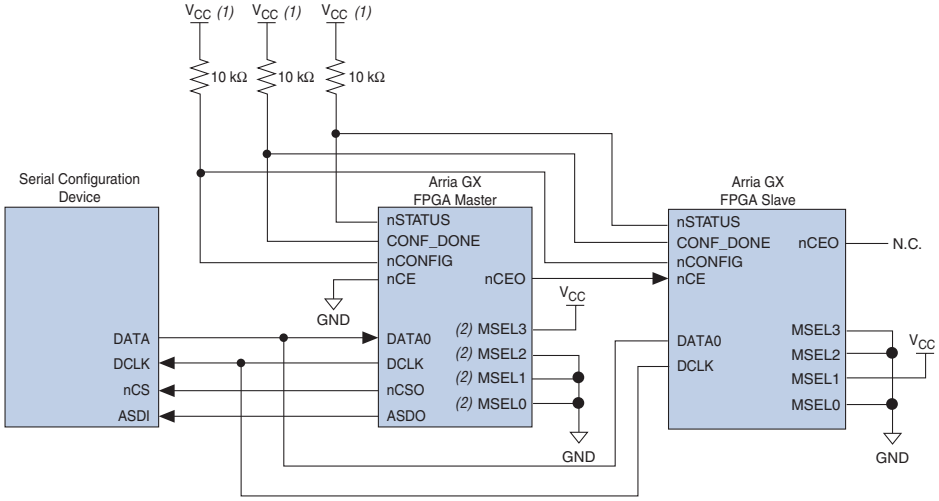
time-out period (maximum of 100 µs), and retries configuration. If this option is turned off, the system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 2 µs to restart configuration.

When the Arria GX device is in user mode, you can initiate reconfiguration by pulling the nCONFIG pin low. The nCONFIG pin should be low for at least 2 µs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Once nCONFIG returns to a logic high level and nSTATUS is released by the Arria GX device, reconfiguration begins.

You can configure multiple Arria GX devices using a single serial configuration device. You can cascade multiple Arria GX devices using the chip-enable (nCE) and chip-enable-out (nCEO) pins. The first device in the chain must have its nCE pin connected to ground. You must connect its nCEO pin to the nCE pin of the next device in the chain. When the first device captures all of its configuration data from the bitstream, it drives the nCEO pin low, enabling the next device in the chain. You must leave the nCEO pin of the last device unconnected. The nCONFIG, nSTATUS, CONF_DONE, DCLK, and DATA0 pins of each device in the chain are connected (refer to Figure 11–13).

This first Arria GX device in the chain is the configuration master and controls configuration of the entire chain. You must connect its MSEL pins to select the AS configuration scheme. The remaining Arria GX devices are configuration slaves and you must connect their MSEL pins to select the PS configuration scheme. Any other Altera device that supports PS configuration can also be part of the chain as a configuration slave. Figure 11–13 shows the pin connections for this setup.

Figure 11–13. Multi-Device AS Configuration



Notes to Figure 11–13:

(1) Connect the pull-up resistors to a 3.3-V supply.

(2) If using an EPCS4 device, `MSEL[3..0]` should be set to **1101**. Refer to Table 11–10 on page 11–33 for more details.

As shown in Figure 11–13, the `nSTATUS` and `CONF_DONE` pins on all target devices are connected together with external pull-up resistors. These pins are open-drain bidirectional pins on the devices. When the first device asserts `nCEO` (after receiving all of its configuration data), it releases its `CONF_DONE` pin. But the subsequent devices in the chain keep this shared `CONF_DONE` line low until they have received their configuration data. When all target devices in the chain have received their configuration data and have released `CONF_DONE`, the pull-up resistor drives a high level on this line and all devices simultaneously enter initialization mode.

If an error occurs at any point during configuration, the `nSTATUS` line is driven low by the failing device. If you enable the Auto-restart configuration after error option, reconfiguration of the entire chain begins after a reset time-out period (a maximum of 100 µs). If the **Auto-restart configuration after error** option is turned off, the external system must monitor `nSTATUS` for errors and then pulse `nCONFIG` low to restart configuration. The external system can pulse `nCONFIG` if it is under system control rather than tied to V_{CC}.

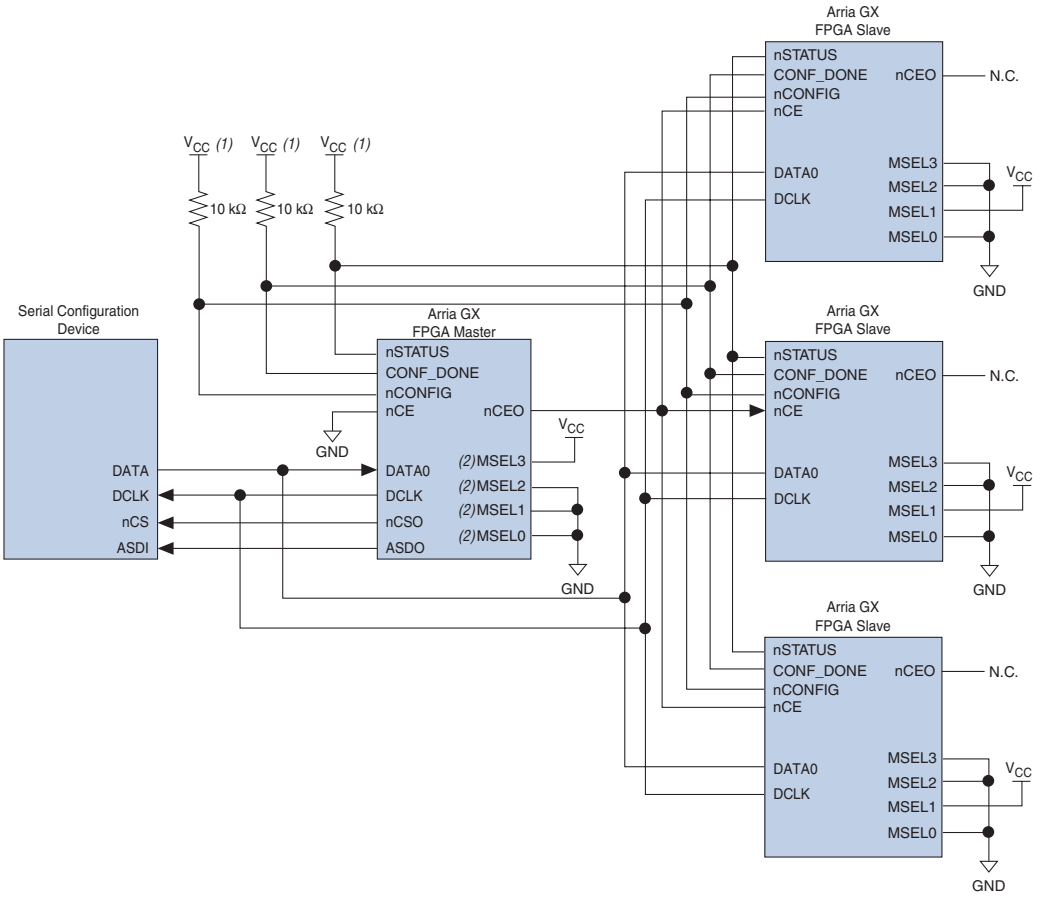
While you can cascade Arria GX devices, serial configuration devices cannot be cascaded or chained together.

If the configuration bitstream size exceeds the capacity of a serial configuration device, you must select a larger configuration device and/or enable the compression feature. When configuring multiple devices, the size of the bitstream is the sum of the individual devices' configuration bitstreams.

A system may have multiple devices that contain the same configuration data. In active serial chains, this can be implemented by storing two copies of the SOF in the serial configuration device. The first copy would configure the master Arria GX device; the second copy would configure all remaining slave devices concurrently. All slave devices must be the same density and package. The setup is similar to Figure 11–13, where the master is set up in active serial mode and the slave devices are set up in passive serial mode.

To configure four identical Arria GX devices with the same SOF, you could set up the chain similar to the example shown in Figure 11–14. The first device is the master device and its MSEL pins should be set to select AS configuration. The other three slave devices are set up for concurrent configuration and its MSEL pins should be set to select PS configuration. The nCEO pin from the master device drives the nCE input pins on all three slave devices, and the DATA and DCLK pins connect in parallel to all four devices. During the first configuration cycle, the master device reads its configuration data from the serial configuration device while holding nCEO high. After completing its configuration cycle, the master drives nCE low and transmits the second copy of the configuration data to all three slave devices, configuring them simultaneously.

Figure 11–14. Multi-Device AS Configuration When devices Receive the Same Data



Notes to Figure 11–14:

(1) Connect the pull-up resistors to a 3.3-V supply.

(2) If using an EPCS4 device, `MSEL[3..0]` should be set to **1101**. Refer to Table 11–10 on page 11–33 for more details.

Estimating Active Serial Configuration Time

Active serial configuration time is dominated by the time it takes to transfer data from the serial configuration device to the Arria GX device. This serial interface is clocked by the Arria GX DCLK output (generated from an internal oscillator). As listed in Table 11–11 on page 11–35, the DCLK minimum frequency when choosing to use the 40-MHz oscillator is 20 MHz (50 ns). Therefore, the maximum configuration time estimate for an EP2S15 device (5 MBits of uncompressed data) is:

RBF Size (minimum DCLK period / 1 bit per DCLK cycle) = estimated maximum configuration time

5 Mbits × (50 ns / 1 bit) = 250 ms

To estimate the typical configuration time, use the typical DCLK period as listed in Table 11–11. With a typical DCLK period of 38.46 ns, the typical configuration time is 192 ms. Enabling compression reduces the amount of configuration data that is transmitted to the Arria GX device, which also reduces configuration time. On average, compression reduces configuration time by 50%.

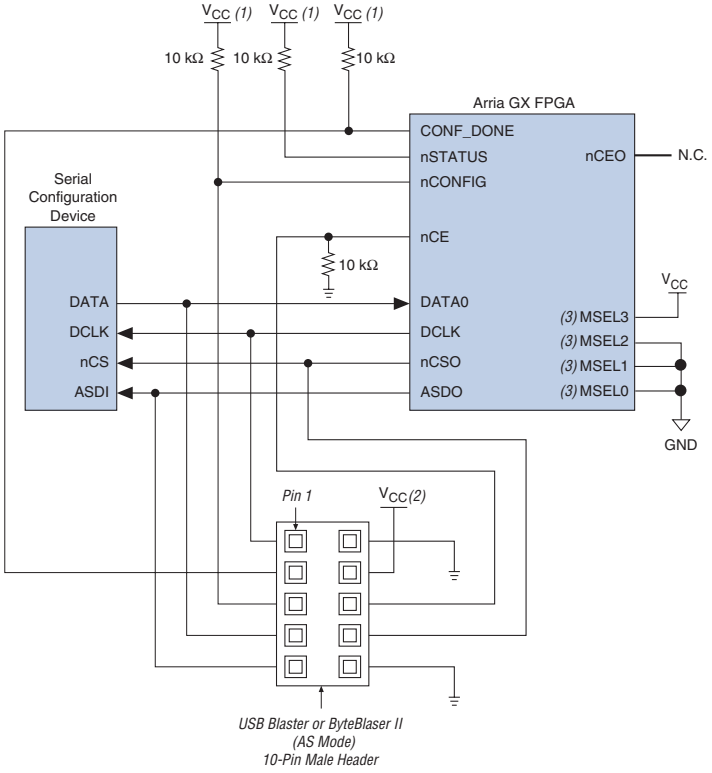
Programming Serial Configuration Devices

Serial configuration devices are non-volatile, flash-memory-based devices. You can program these devices in-system using the USB-Blaster™ or ByteBlaster™ II download cable. Alternatively, you can program them using the Altera Programming Unit (APU), supported third-party programmers, or a microprocessor with the SRunner software driver.

You can perform in-system programming of serial configuration devices via the AS programming interface. During in-system programming, the download cable disables device access to the AS interface by driving the nCE pin high. Arria GX devices are also held in reset by a low level on nCONFIG. After programming is complete, the download cable releases nCE and nCONFIG, allowing the pull-down and pull-up resistors to drive GND and V_{CC}, respectively. Figure 11–15 shows the download cable connections to the serial configuration device.

For more information about the USB Blaster download cable, refer to the *USB-Blaster USB Port Download Cable User Guide*. For more information about the ByteBlaster II cable, refer to the *ByteBlaster II Download Cable User Guide*.

Figure 11–15. In-System Programming of Serial Configuration Devices



Notes to Figure 11–15:

(1) Connect these pull-up resistors to 3.3-V supply.

(2) Power up the ByteBlaster II cable's V_{CC} with a 3.3-V supply.

(3) If using an EPCS4 device, `MSEL[3..0]` should be set to **1101**. Refer to Table 11–10 on page 11–33 for more details.

You can program serial configuration devices with the Quartus II software with the Altera programming hardware and the appropriate configuration device programming adapter. The EPCS1 and EPCS4 devices are offered in an eight-pin small outline integrated circuit (SOIC) package.

In production environments, serial configuration devices can be programmed using multiple methods. Altera programming hardware or other third-party programming hardware can be used to program blank serial configuration devices before they are mounted onto printed circuit

boards (PCBs). Alternatively, you can use an on-board microprocessor to program the serial configuration device in-system using C-based software drivers provided by Altera.

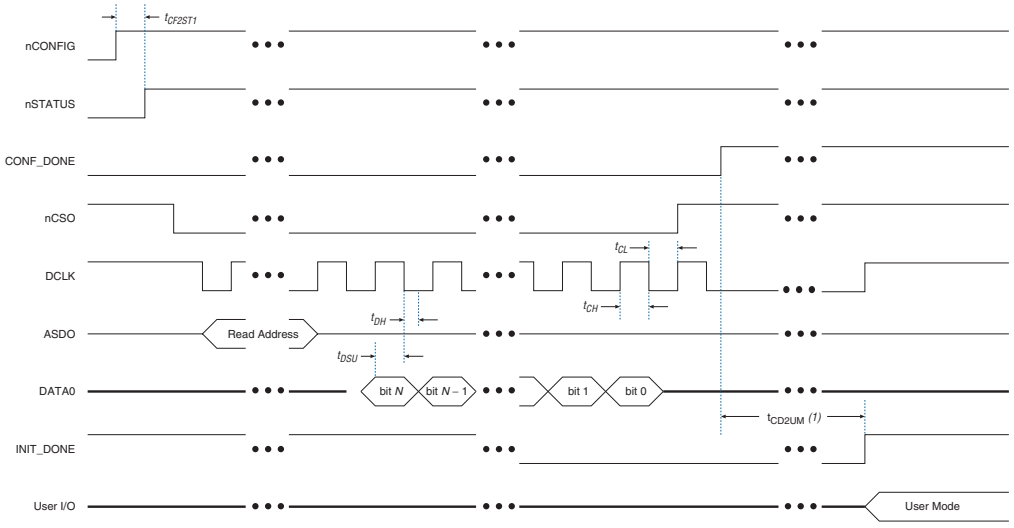
A serial configuration device can be programmed in-system by an external microprocessor using SRunner. SRunner is a software driver developed for embedded serial configuration device programming, which can be easily customized to fit in different embedded systems. SRunner is able to read a raw programming data (.**rpd**) file and write to the serial configuration devices. The serial configuration device programming time using SRunner is comparable to the programming time with the Quartus II software.

For more information about SRunner, refer to *AN 418: SRunner: An Embedded Solution for Serial Configuration* and the source code on the Altera web site at www.altera.com.

For more information on programming serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook*.

Figure 11–16 shows the timing waveform for the AS configuration scheme using a serial configuration device.

Figure 11–16. AS Configuration Timing



Note to Figure 11–16:
(1) The initialization clock can come from the Arria GX device's internal oscillator or the CLKUSR pin.

Table 11–12 shows the AS timing parameters for Arria GX devices.

Table 11–12. AS Timing Parameters for Arria GX Devices

Symbol	Parameter	Condition	Minimum	Typical	Maximum
t_{CF2ST1}	nCONFIG high to nSTATUS high				100
t_{DSU}	Data setup time before falling edge on DCLK		7		
t_{DH}	Data hold time after falling edge on DCLK		0		
t_{CH}	DCLK high time		10		
t_{CL}	DCLK low time		10		
t_{CD2UM}	CONF_DONE high to user mode		20		100

Passive Serial Configuration

PS configuration of Arria GX devices can be performed using an intelligent host, such as a MAX II device or microprocessor with flash memory, an Altera configuration device, or a download cable. In the PS scheme, an external host (MAX II device, embedded processor, configuration device, or host PC) controls configuration. Configuration data is clocked into the target Arria GX device via the DATA0 pin at each rising edge of DCLK.

The Arria GX decompression feature is fully available when configuring your Arria GX device using PS mode.

Table 11–13 shows the MSEL pin settings when using the PS configuration scheme.

Table 11–13. Arria GX MSEL Pin Settings for PS Configuration Schemes

Configuration Scheme	MSEL3	MSEL2	MSEL1	MSEL0
PS	0	0	1	0
PS when using Remote System Upgrade *(1)*	0	1	1	0

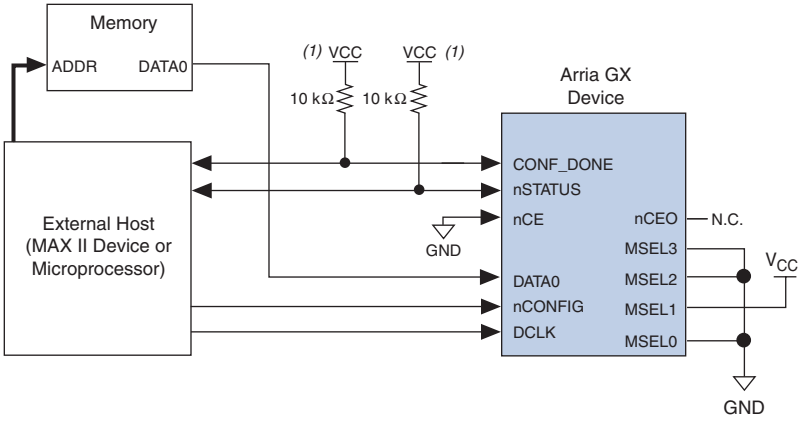
Note to Table 11–13:

(1) This scheme requires that you drive the RUnLU pin to specify either remote update or local update. For more information about remote system upgrade in Arria GX devices, refer to the *Remote System Upgrades with Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*.

PS Configuration Using a MAX II Device as an External Host

In the PS configuration scheme, a MAX II device can be used as an intelligent host that controls the transfer of configuration data from a storage device, such as flash memory, to the target Arria GX device. Configuration data can be stored in RBF, HEX, or TTF format. Figure 11–17 shows the configuration interface connections between a Arria GX device and a MAX II device for single device configuration.

Figure 11–17. Single Device PS Configuration Using an External Host



Note to Figure 11–17:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for the device. V_{CC} should be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

Upon power-up, Arria GX devices go through a POR. The POR delay is dependent on the PORSEL pin setting: when PORSEL is driven low, the POR time is approximately 100 ms; when PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device resets, holds nSTATUS low, and tri-states all user I/O pins. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *Arria GX Device Handbook*.

The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration, the MAX II device must generate a low-to-high transition on the nCONFIG pin.

V_{CCINT}, V_{CCIO}, and V_{CCPD} of the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. Once nSTATUS is released, the device is ready to receive configuration data and the configuration stage begins. When nSTATUS is pulled high, the MAX II device should place the configuration data, one bit at a time, on the DATA0 pin. If you are using configuration data in RBF, HEX, or TTF format, you must send the least significant bit (LSB) of each data byte first. For example, if the RBF contains the byte sequence 02 1B EE 01 FA, the serial bitstream you should transmit to the device is 0100-0000 1101-1000 0111-0111 1000-0000 0101-1111.

Arria GX devices receive configuration data on the DATA0 pin and the clock is received on the DCLK pin. Data is latched into the device on the rising edge of DCLK. Data is continuously clocked into the target device until CONF_DONE goes high. After the device has received all configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by an external 10-kΩ pull-up resistor. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

In Arria GX devices, the initialization clock source is either the internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Arria GX device provides itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device. Driving DCLK to the device after configuration is complete does not affect device operation.

You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. The **Enable user-supplied start-up clock (CLKUSR)** option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR will not affect the configuration process. After all configuration data has been accepted and CONF_DONE

goes high, CLKUSR will be enabled after the time specified as t_{CD2CU}. After this time period elapses, Arria GX devices require 299 clock cycles to initialize properly and enter user mode. Arria GX devices support a CLKUSR f_{MAX} of 100 MHz.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used, it will be high due to an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low. When initialization is complete, the INIT_DONE pin will be released and pulled high. The MAX II device must be able to detect this low-to-high transition which signals the device has entered user mode. When initialization is complete, the device enters user mode. In user-mode, the user I/O pins will no longer have weak pull-up resistors and will function as assigned in your design.

To ensure DCLK and DATA0 are not left floating at the end of configuration, the MAX II device must drive them either high or low, whichever is convenient on your board. The DATA[0] pin is available as a user I/O pin after configuration. When the PS scheme is chosen in the Quartus II software, as a default, this I/O pin is tri-stated in user mode and should be driven by the MAX II device. To change this default option in the Quartus II software, select the **Dual-Purpose Pins** tab of the **Device & Pin Options** dialog box.

The configuration clock (DCLK) speed must be below the specified frequency to ensure correct configuration. No maximum DCLK period exists, which means you can pause configuration by halting DCLK for an indefinite amount of time.

If an error occurs during configuration, the device drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the MAX II device that there is an error. If the **Auto-restart configuration after error** option (available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box) is turned on, the Arria GX device releases nSTATUS after a reset time-out period (maximum of 100 µs). After nSTATUS is released and pulled high by a pull-up resistor, the MAX II device can try to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

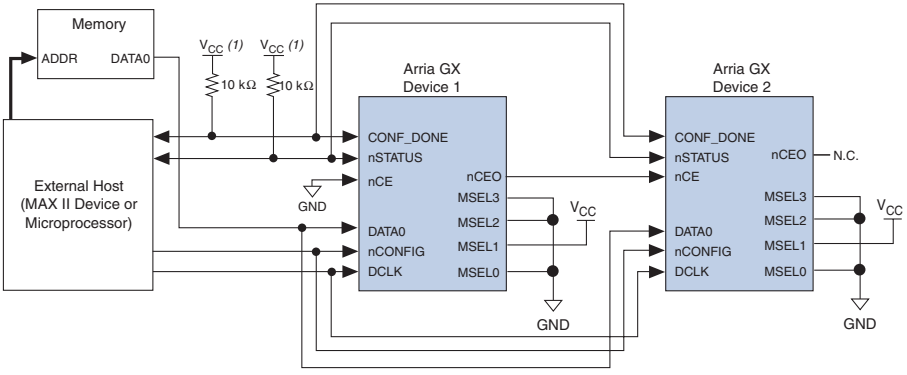
The MAX II device can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The CONF_DONE pin must be monitored by the MAX II device to detect errors and determine when programming completes. If all configuration data is sent, but CONF_DONE or INIT_DONE have not gone high, the MAX II device must reconfigure the target device.

If the optional CLKUSR pin is being used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure that CLKUSR continues toggling during the time nSTATUS is low (maximum of 100 µs).

When the device is in user-mode, you can initiate a reconfiguration by transitioning the nCONFIG pin low to high. The nCONFIG pin must be low for at least 2 µs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Once nCONFIG returns to a logic high level and nSTATUS is released by the device, reconfiguration begins.

Figure 11–18 shows how to configure multiple devices using a MAX II device. This circuit is similar to the PS configuration circuit for a single device, except Arria GX devices are cascaded for multi-device configuration.

Figure 11–18. Multi-Device PS Configuration Using an External Host



Note to Figure 11–18:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain. V_{CC} should be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

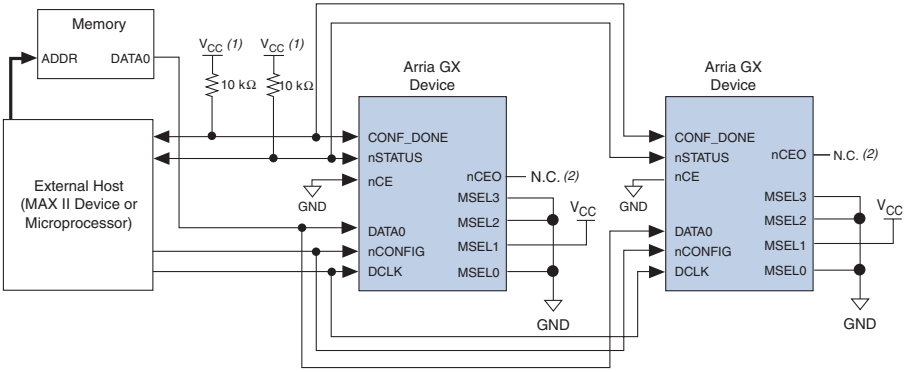
In multi-device PS configuration, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration within one clock cycle. Therefore, the transfer of data destinations is transparent to the MAX II device. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Because all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

If the **Auto-restart configuration after error** option is turned on, the devices release their nSTATUS pins after a reset time-out period (maximum of 100 µs). After all nSTATUS pins are released and pulled high, the MAX II device can try to reconfigure the chain without needing to pulse nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

In your system, you can have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete configuration at the same time. Figure 11–19 shows multi-device PS configuration when both Arria GX devices are receiving the same configuration data.

Figure 11–19. Multiple-Device PS Configuration When Both Devices Receive the Same Data



Notes to Figure 11–19:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain. V_{CC} should be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

(2) The nCEO pins of both devices are left unconnected when configuring the same configuration data into multiple devices.

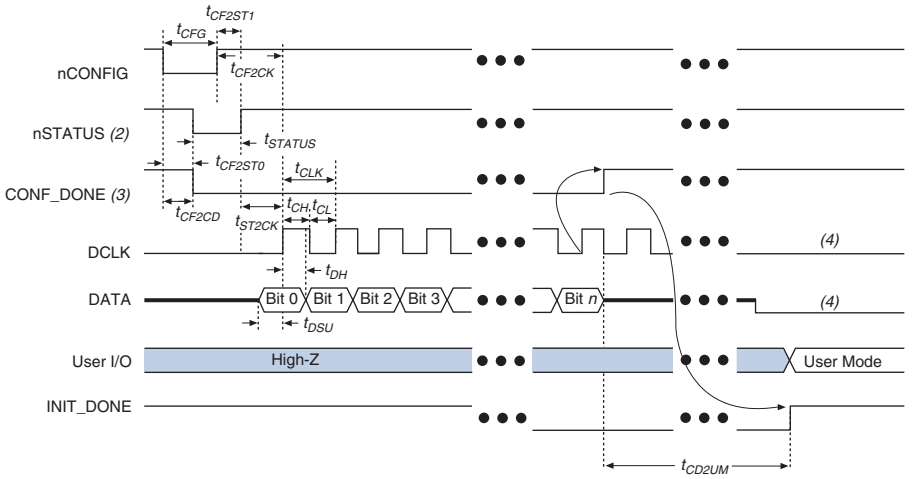
You can use a single configuration chain to configure Arria GX devices with other Altera devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information about configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook*.

PS Configuration Timing

Figure 11–20 shows the timing waveform for PS configuration when using a MAX II device as an external host.

Figure 11–20. PS Configuration Timing Waveform *Note (1)*



Notes to Figure 11–20:

(1) The beginning of this waveform shows the device in user-mode. In user-mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.
(2) Upon power-up, the Arria GX device holds nSTATUS low for the time of the POR delay.
(3) Upon power-up, before and during configuration, CONF_DONE is low.
(4) DCLK should not be left floating after configuration. It should be driven high or low, whichever is more convenient. DATA[0] is available as a user I/O pin after configuration and the state of this pin depends on the dual-purpose pin settings.

Table 11–14 defines the timing parameters for Arria GX devices for PS configuration.

Table 11–14. PS Timing Parameters for Arria GX Devices (Part 1 of 2) *Note (1)*

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		800	ns
t_{CFG}	nCONFIG low pulse width	2		µs
t_{STATUS}	nSTATUS low pulse width	10	100 (2)	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high		100 (2)	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	100		µs
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2		µs
t_{DSU}	Data setup time before rising edge on DCLK	5		ns
t_{DH}	Data hold time after rising edge on DCLK	0		ns

Table 11–14. PS Timing Parameters for Arria GX Devices (Part 2 of 2) *Note (1)*

Symbol	Parameter	Min	Max	Units
t_{CH}	DCLK high time	4		ns
t_{CL}	DCLK low time	4		ns
t_{CLK}	DCLK period	10		ns
f_{MAX}	DCLK frequency		100	MHz
t_R	Input rise time		40	ns
t_F	Input fall time		40	ns
t_{CD2UM}	CONF_DONE high to user mode *(3)*	20	100	µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period		
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	t_{CD2CU} + (299 × CLKUSR period)		

Notes to Table 11–14:

(1) This information is preliminary.

(2) This value is applicable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(3) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting the device.

Device configuration options and how to create configuration files are discussed further in the *Software Settings* section in volume 2 of the *Configuration Handbook*.

An example PS design that uses a MAX II device as the external host for configuration will be available when devices are available.

PS Configuration Using a Microprocessor

In the PS configuration scheme, a microprocessor can control the transfer of configuration data from a storage device, such as flash memory, to the target Arria GX device.

All information in the "PS Configuration Using a MAX II Device as an External Host" on page 11–45 section is also applicable when using a microprocessor as an external host. Refer to that section for all configuration and timing information.

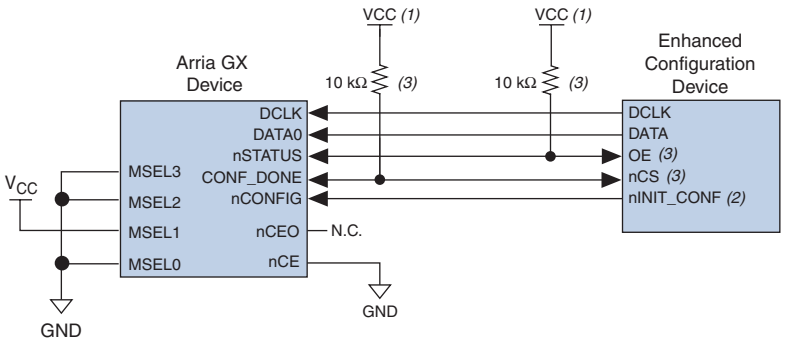
PS Configuration Using a Configuration Device

You can use an Altera configuration device, such as an enhanced configuration device or EPC2 device, to configure Arria GX devices using a serial configuration bitstream. Configuration data is stored in the configuration device. Figure 11–21 shows the configuration interface connections between an Arria GX device and a configuration device.

The figures in this chapter only show the configuration-related pins and the configuration pin connections between the configuration device and the device.

For more information on the enhanced configuration device and flash interface pins (such as PGM[2..0], EXCLK, PORSEL, A[20..0], and DQ[15..0]), refer to the *Enhanced Configuration Devices (EPC4, EPC8, & EPC16) Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

Figure 11–21. Single Device PS Configuration Using an Enhanced Configuration Device



Notes to Figure 11–21:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

The value of the internal pull-up resistors on the enhanced configuration devices and EPC2 devices can be found in the Operating Conditions table of the *Enhanced Configuration Devices (EPC4, EPC8, & EPC16) Data Sheet* chapter or the *Configuration Devices for SRAM-based LUT Devices Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

When using enhanced configuration devices or EPC2 devices, nCONFIG of the device can be connected to nINIT_CONF of the configuration device, which allows the INIT_CONF JTAG instruction to initiate device configuration. The nINIT_CONF pin does not need to be connected if its functionality is not used. An internal pull-up resistor on the nINIT_CONF pin is always active in enhanced configuration devices and EPC2 devices, which means an external pull-up resistor should not be used if nCONFIG is tied to nINIT_CONF.

Upon power-up, the Arria GX devices go through a POR. The POR delay is dependent on the PORSEL pin setting. When PORSEL is driven low, the POR time is approximately 100 ms. If PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device will reset, hold nSTATUS low, and tri-state all user I/O pins. The configuration device also goes through a POR delay to allow the power supply to stabilize. The POR time for EPC2 devices is 200 ms (maximum). The POR time for enhanced configuration devices can be set to either 100 ms or 2 ms, depending on its PORSEL pin setting. If the PORSEL pin is connected to GND, the POR delay is 100 ms. If the PORSEL pin is connected to V_{CC}, the POR delay is 2 ms. During this time, the configuration device drives its OE pin low. This low signal delays configuration because the OE pin is connected to the target device's nSTATUS pin.

When selecting a POR time, you need to ensure that the device completes power-up before the enhanced configuration device exits POR. Altera recommends that you choose a POR time for the Arria GX device of 12 ms, while selecting a POR time for the enhanced configuration device of 100 ms.

When both devices complete POR, they release their open-drain OE or nSTATUS pin, which is then pulled high by a pull-up resistor. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *DC & Switching Characteristics* chapter in volume 1 of the *Arria GX Device Handbook*.

When the power supplies have reached the appropriate operating voltages, the target device senses the low-to-high transition on nCONFIG and initiates the configuration cycle. The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. The beginning of configuration can be delayed by holding the nCONFIG or nSTATUS pin low.

To begin configuration, power the V_{CCINT}, V_{CCIO}, and V_{CCPD} voltages (for the banks where the configuration and JTAG pins reside) to the appropriate voltage levels.

When nCONFIG goes high, the device comes out of reset and releases the nSTATUS pin, which is pulled high by a pull-up resistor. Enhanced configuration and EPC2 devices have an optional internal pull-up resistor on the OE pin. This option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If this internal pull-up resistor is not used, an external 10-kΩ pull-up resistor on the OE-nSTATUS line is required. Once nSTATUS is released, the device is ready to receive configuration data and the configuration stage begins.

When nSTATUS is pulled high, OE of the configuration device also goes high and the configuration device clocks data out serially to the device using the Arria GX device's internal oscillator. Arria GX devices receive configuration data on the DATA0 pin and the clock is received on the DCLK pin. Data is latched into the device on the rising edge of DCLK.

After the device has received all the configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by a pull-up resistor. Because CONF_DONE is tied to the configuration device's nCS pin, the configuration device is disabled when CONF_DONE goes high. Enhanced configuration and EPC2 devices have an optional internal pull-up resistor on the nCS pin. This option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If this internal pull-up resistor is not used, an external 10-kΩ pull-up resistor on the nCS-CONF_DONE line is required. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin.

In Arria GX devices, the initialization clock source is either the internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If you are using internal oscillator, the Arria GX device supplies itself with enough clock cycles for proper initialization. You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. You can turn on the **Enable user-supplied start-up clock** (**CLKUSR**) option in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR will not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, CLKUSR will be enabled after the time specified as t_{CD2CU}. After this time period elapses, the Arria GX devices require 299 clock cycles to initialize properly and enter user mode. Arria GX devices support a CLKUSR f_{MAX} of 100 MHz.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If you are using the INIT_DONE pin, it will be high due to an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. This low-to-high transition signals that the device has entered user mode. In user-mode, the user I/O pins will no longer have weak pull-up resistors and will function as assigned in your design. Enhanced configuration devices and EPC2 devices drive DCLK low and DATA0 high at the end of configuration.

If an error occurs during configuration, the device drives its nSTATUS pin low, resetting itself internally. Because the nSTATUS pin is tied to OE, the configuration device will also be reset. If the **Auto-restart configuration after error** option, available in the Quartus II software, from the **General** tab of the **Device & Pin Options** dialog box is turned on, the device automatically initiates reconfiguration if an error occurs. The Arria GX devices release the nSTATUS pin after a reset time-out period (maximum of 100 µs). When the nSTATUS pin is released and pulled high by a pull-up resistor, the configuration device reconfigures the chain. If this option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 2 µs to restart configuration. The external system can pulse nCONFIG if nCONFIG is under system control rather than tied to V_{CC}.

In addition, if the configuration device sends all of its data and then detects that CONF_DONE has not gone high, it recognizes that the device has not configured successfully. Enhanced configuration devices wait for 64 DCLK cycles after the last configuration bit was sent for CONF_DONE to reach a high state. EPC2 devices wait for 16 DCLK cycles. In this case, the configuration device pulls its OE pin low, driving the target device's nSTATUS pin low. If the **Auto-restart configuration after error** option is set in the software, the target device resets and then releases its nSTATUS pin after a reset time-out period (maximum of 100 µs). When nSTATUS returns to a logic high level, the configuration device tries to reconfigure the device.

When CONF_DONE is sensed low after configuration, the configuration device recognizes that the target device has not configured successfully. Therefore, your system should not pull CONF_DONE low to delay initialization. Instead, use the CLKUSR option to synchronize the

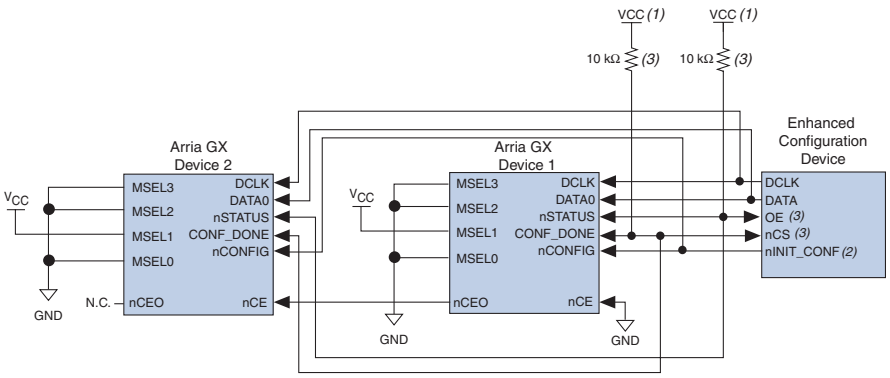
initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain will initialize together if their CONF_DONE pins are tied together.

If you are using the optional CLKUSR pin and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure that CLKUSR continues toggling during the time nSTATUS is low (maximum of 100 µs).

When the device is in user-mode, pulling the nCONFIG pin low initiates a reconfiguration. The nCONFIG pin should be low for at least 2 µs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Because CONF_DONE is pulled low, this activates the configuration device because it sees its nCS pin drive low. Once nCONFIG returns to a logic high level and nSTATUS is released by the device, reconfiguration begins.

Figure 11–22 shows how to configure multiple devices with an enhanced configuration device. This circuit is similar to the configuration device circuit for a single device, except Arria GX devices are cascaded for multi-device configuration.

Figure 11–22. Multi-Device PS Configuration Using an Enhanced Configuration Device



Notes to Figure 11–22:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

Enhanced configuration devices cannot be cascaded.

When performing multi-device configuration, you must generate the configuration device's POF from each project's SOF. You can combine multiple SOFs using the **Convert Programming Files** window in the Quartus II software.

For more information about creating configuration files for multi-device configuration chains, refer to the *Software Settings* section in volume 2 of the *Configuration Handbook*.

In multi-device PS configuration, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, prompting the second device to begin configuration. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device.

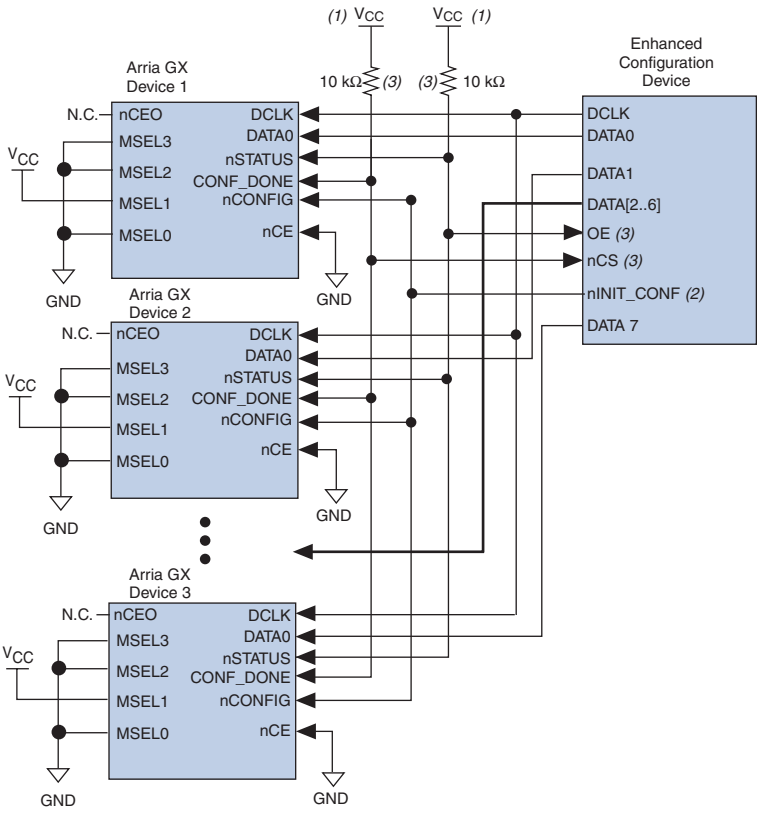
When configuring multiple devices, configuration does not begin until all devices release their OE or nSTATUS pins. Similarly, because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Because all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This low signal drives the OE pin low on the enhanced configuration device and drives nSTATUS low on all devices, causing them to enter a reset state. This behavior is similar to a single device detecting an error.

If the **Auto-restart configuration after error** option is turned on, the devices will automatically initiate reconfiguration if an error occurs. The devices will release their nSTATUS pins after a reset time-out period (maximum of 100 µs). When all the nSTATUS pins are released and pulled high, the configuration device tries to reconfigure the chain. If the **Auto-restart configuration after error** option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 2 µs to restart configuration. The external system can pulse nCONFIG if nCONFIG is under system control rather than tied to V_{CC}.

The enhanced configuration devices also support parallel configuration of up to eight devices. The n-bit (n = 1, 2, 4, or 8) PS configuration mode allows enhanced configuration devices to concurrently configure devices or a chain of devices. In addition, these devices do not have to be the same device family or density as they can be any combination of Altera devices. An individual enhanced configuration device DATA line is available for each targeted device. Each DATA line can also feed a daisy chain of devices. Figure 11–23 shows how to concurrently configure multiple devices using an enhanced configuration device.

Figure 11–23. Concurrent PS Configuration of Multiple Devices Using an Enhanced Configuration Device



Notes to Figure 11–23:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

The Quartus II software only allows the selection of n-bit PS configuration modes, where n must be 1, 2, 4, or 8. However, you can use these modes to configure any number of devices from 1 to 8. When configuring SRAM-based devices using n-bit PS modes, use Table 11–15 to select the appropriate configuration mode for the fastest configuration times.

Table 11–15. Recommended Configuration Using n-Bit PS Modes

Number of Devices *(1)*	Recommended Configuration Mode
1	1-bit PS
2	2-bit PS
3	4-bit PS
4	4-bit PS
5	8-bit PS
6	8-bit PS
7	8-bit PS
8	8-bit PS

Note to Table 11–15:

(1) Assume that each DATA line is only configuring one device, not a daisy chain of devices.

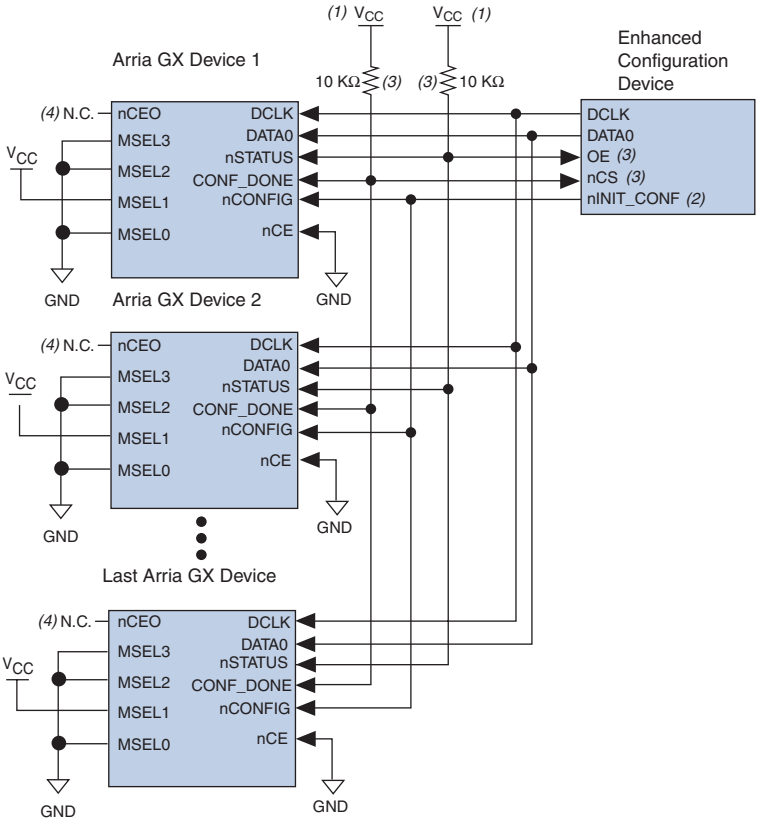
For example, if you configure three devices, you would use the 4-bit PS mode. For the DATA0, DATA1, and DATA2 lines, the corresponding SOF data is transmitted from the configuration device to the device. For DATA3, you can leave the corresponding Bit3 line blank in the Quartus II software. On the PCB, leave the DATA3 line from the enhanced configuration device unconnected.

Alternatively, you can daisy chain two devices to one DATA line while the other DATA lines drive one device each. For example, you could use the 2-bit PS mode to drive two devices with DATA Bit0 (two EP2S15 devices) and the third device (EP2S30 device) with DATA Bit1. This 2-bit PS configuration scheme requires less space in the configuration flash memory, but can increase the total system configuration time.

A system may have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete

configuration at the same time. Figure 11–24 shows multi-device PS configuration when the Arria GX devices are receiving the same configuration data.

Figure 11–24. Multiple-Device PS Configuration Using an Enhanced Configuration Device When Devices Receive the Same Data



Notes to Figure 11–24:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

(4) The nCEO pins of all devices are left unconnected when configuring the same configuration data into multiple devices.

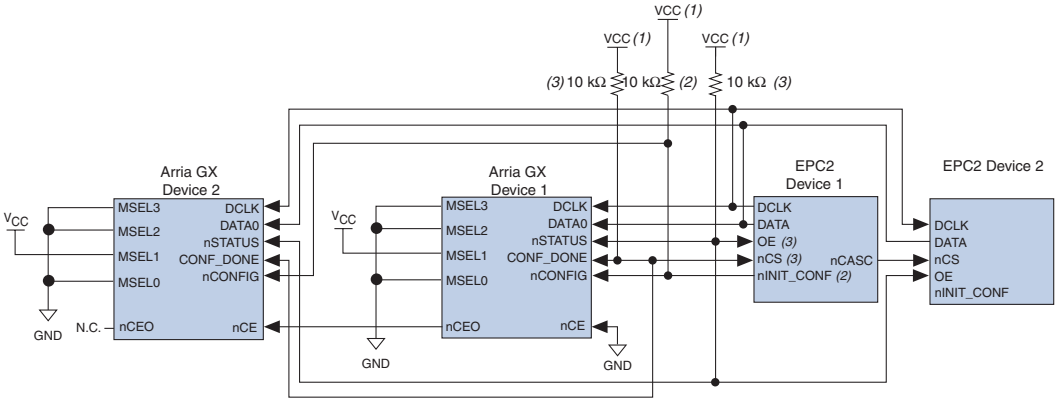
You can cascade several EPC2 devices to configure multiple Arria GX devices. The first configuration device in the chain is the master configuration device, while the subsequent devices are the slave devices. The master configuration device sends DCLK to the Arria GX devices and to the slave configuration devices. The first EPC device's nCS pin is connected to the CONF_DONE pins of the devices, while its nCASC pin is connected to nCS of the next configuration device in the chain. The last device's nCS input comes from the previous device, while its nCASC pin is left floating. When all data from the first configuration device is sent, it drives nCASC low, which in turn drives nCS on the next configuration device. A configuration device requires less than one clock cycle to activate a subsequent configuration device, so the data stream is uninterrupted.

Enhanced configuration devices cannot be cascaded.

Because all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, the master configuration device stops configuration for the entire chain and the entire chain must be reconfigured. For example, if the master configuration device does not detect CONF_DONE going high at the end of configuration, it resets the entire chain by pulling its OE pin low. This low signal drives the OE pin low on the slave configuration device(s) and drives nSTATUS low on all devices, causing them to enter a reset state. This behavior is similar to the device detecting an error in the configuration data.

Figure 11–25 shows how to configure multiple devices using cascaded EPC2 devices.

Figure 11–25. Multi-Device PS Configuration Using Cascaded EPC2 Devices



Notes to Figure 11–25:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used.

(3) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external 10-kΩ pull-up resistors should not be used. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

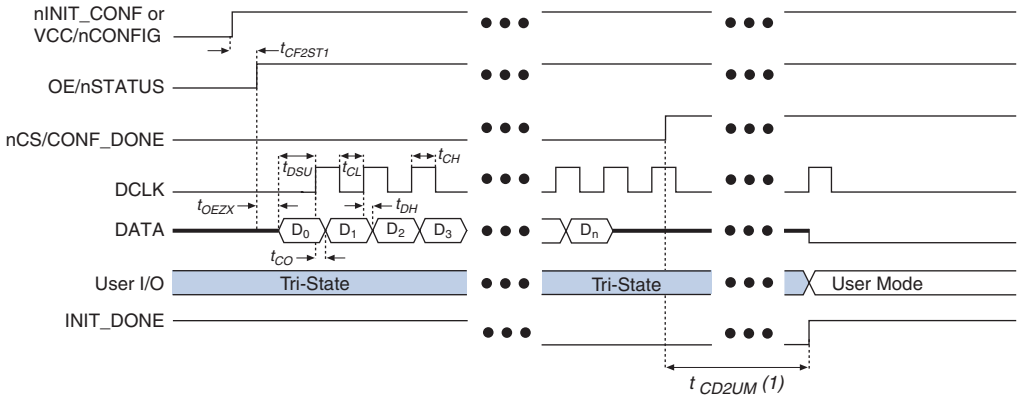
When using enhanced configuration devices or EPC2 devices, nCONFIG of the device can be connected to nINIT_CONF of the configuration device, allowing the INIT_CONF JTAG instruction to initiate device configuration. The nINIT_CONF pin does not need to be connected if its functionality is not used. An internal pull-up resistor on the nINIT_CONF pin is always active in the enhanced configuration devices and the EPC2 devices, which means that you shouldn't be using an external pull-up resistor if nCONFIG is tied to nINIT_CONF. If you are using multiple EPC2 devices to configure a Arria GX device(s), only the first EPC2 has its nINIT_CONF pin tied to the device's nCONFIG pin.

You can use a single configuration chain to configure Arria GX devices with other Altera devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information on configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook*.

Figure 11–26 shows the timing waveform for the PS configuration scheme using a configuration device.

Figure 11–26. Arria GX PS Configuration Using a Configuration Device Timing Waveform



Note to Figure 11–26:
(1) The initialization clock can come from the Arria GX device's internal oscillator or the **CLKUSR** pin.

For timing information, refer to the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* chapter or the *Configuration Devices for SRAM-Based LUT Devices Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

Device configuration options and how to create configuration files are discussed further in the *Software Settings* chapter of the *Configuration Handbook*.

PS Configuration Using a Download Cable

In this section, the generic term "download cable" includes the Altera USB-Blaster™ universal serial bus (USB) port download cable, MasterBlaster™ serial/USB communications cable, ByteBlaster™ II parallel port download cable, and the ByteBlaster MV parallel port download cable.

In PS configuration with a download cable, an intelligent host (such as a PC) transfers data from a storage device to the device via the USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV cable.

Upon power-up, the Arria GX devices go through a POR. The POR delay is dependent on the PORSEL pin setting. When PORSEL is driven low, the POR time is approximately 100 ms. If PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device will reset, hold nSTATUS low, and tri-state all user I/O pins. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *DC & Switching Characteristics* chapter in volume 1 of the *Arria GX Device Handbook*.

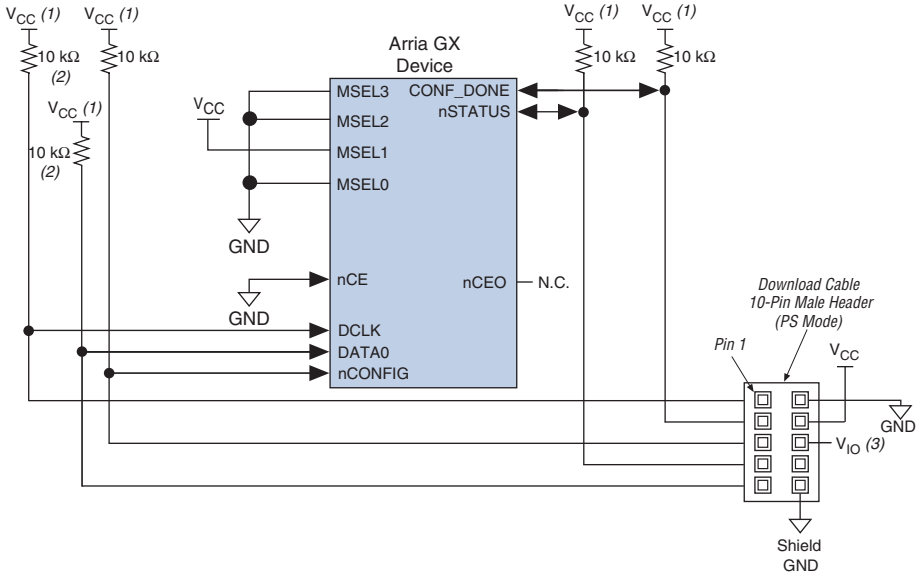
The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration in this scheme, the download cable generates a low-to-high transition on the nCONFIG pin.

To begin configuration, power the V_{CCINT}, V_{CCIO}, and V_{CCPD} voltages (for the banks where the configuration and JTAG pins reside) to the appropriate voltage levels.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. Once nSTATUS is released the device is ready to receive configuration data and the configuration stage begins. The programming hardware or download cable then places the configuration data one bit at a time on the device's DATA0 pin. The configuration data is clocked into the target device until CONF_DONE goes high. The CONF_DONE pin must have an external 10-kΩpull-up resistor in order for the device to initialize.

When using a download cable, setting the **Auto-restart configuration after error** option does not affect the configuration cycle because you must manually restart configuration in the Quartus II software when an error occurs. Additionally, the **Enable user-supplied start-up clock (CLKUSR)** option has no affect on the device initialization because this option is disabled in the SOF when programming the device using the Quartus II programmer and download cable. Therefore, if you turn on the CLKUSR option, you do not need to provide a clock on CLKUSR when you are configuring the device with the Quartus II programmer and a download cable. Figure 11–27 shows PS configuration for Arria GX devices using a USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV cable.

Figure 11–27. PS Configuration Using a USB Blaster, MasterBlaster, ByteBlaster II or ByteBlasterMV Cable



Notes to Figure 11–27:

(1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster (V_{IO} pin), ByteBlaster II, or ByteBlasterMV cable.

(2) The pull-up resistors on DATA0 and DCLK are only needed if the download cable is the only configuration scheme used on your board. This ensures that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, the pull-up resistors on DATA0 and DCLK are not needed.

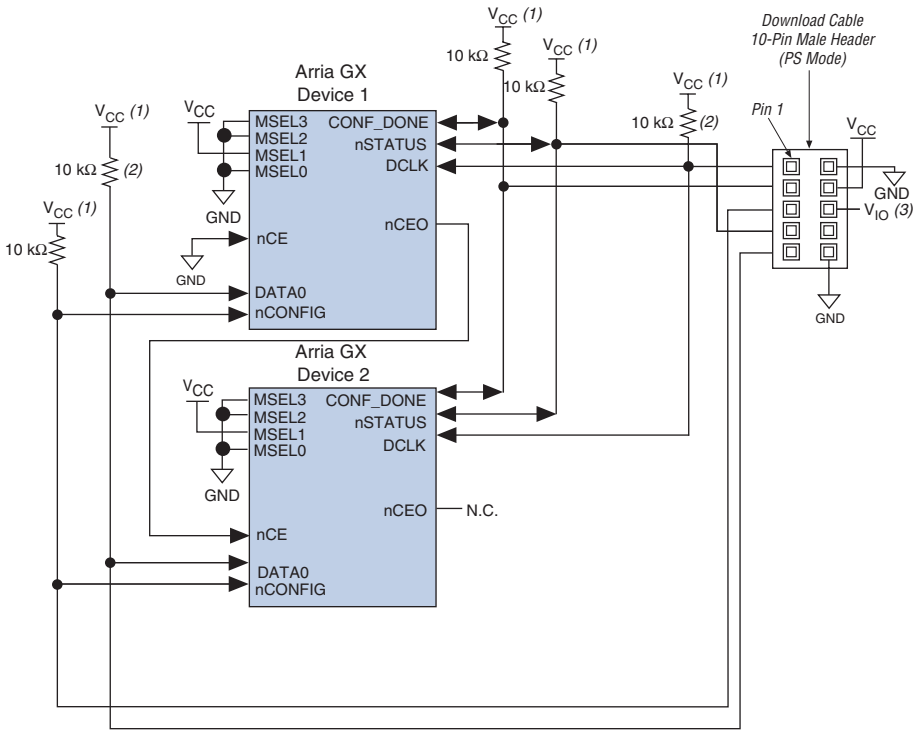
(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. Refer to the *MasterBlaster Serial/USB Communications Cable User Guide* for this value. In the ByteBlasterMV cable, this pin is a no connect. In the USB Blaster and ByteBlaster II cables, this pin is connected to nCE when it is used for active serial programming, otherwise it is a no connect.

You can use a download cable to configure multiple Arria GX devices by connecting each device's nCEO pin to the subsequent device's nCE pin. The first device's nCE pin is connected to GND while its nCEO pin is connected to the nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. All other configuration pins, nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE are connected to every device in the chain. Because all CONF_DONE pins are tied together, all devices in the chain initialize and enter user mode at the same time.

In addition, because the nSTATUS pins are tied together, the entire chain halts configuration if any device detects an error. The **Auto-restart configuration after error** option does not affect the configuration cycle because you must manually restart configuration in the Quartus II software when an error occurs.

Figure 11–28 shows how to configure multiple Arria GX devices with a download cable.

Figure 11–28. Multi-Device PS Configuration using a USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV Cable



Notes to Figure 11–28:

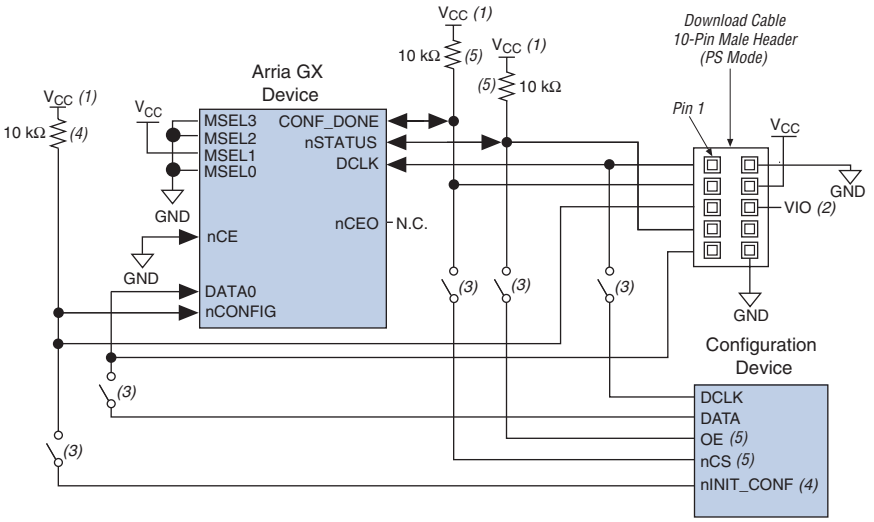
(1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster (V_{IO} pin), ByteBlaster II, or ByteBlasterMV cable.

(2) The pull-up resistors on DATA0 and DCLK are only needed if the download cable is the only configuration scheme used on your board. This is to ensure that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, the pull-up resistors on DATA0 and DCLK are not needed.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. Refer to the *MasterBlaster Serial/USB Communications Cable User Guide* for this value. In the ByteBlasterMV cable, this pin is a no connect. In the USB Blaster and ByteBlaster II cables, this pin is connected to nCE when it is used for active serial programming, otherwise it is a no connect.

If you are using a download cable to configure device(s) on a board that also has configuration devices, electrically isolate the configuration device from the target device(s) and cable. One way of isolating the configuration device is to add logic, such as a multiplexer, that can select between the configuration device and the cable. The multiplexer chip allows bidirectional transfers on the nSTATUS and CONF_DONE signals. Another option is to add switches to the five common signals (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) between the cable and the configuration device. The last option is to remove the configuration device from the board when configuring the device with the cable. Figure 11–29 shows a combination of a configuration device and a download cable to configure an device.

Figure 11–29. PS Configuration with a Download Cable and Configuration Device Circuit



Notes to Figure 11–29:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. Refer to the *MasterBlaster Serial/USB Communications Cable User Guide* for this value. In the ByteBlasterMV cable, this pin is a no connect. In the USB Blaster and ByteBlaster II cables, this pin is connected to nCE when it is used for active serial programming, otherwise it is a no connect.

(3) You should not attempt configuration with a download cable while a configuration device is connected to an Arria GX device. Instead, you should either remove the configuration device from its socket when using the download cable or place a switch on the five common signals between the download cable and the configuration device.

(4) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) has an internal pull-up resistor that is always active. This means an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used.

(5) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-up resistors on configuration device** option when generating programming files.

For more information on how to use the USB Blaster, MasterBlaster, ByteBlaster II or ByteBlasterMV cables, refer to the following data sheets:

- *USB-Blaster Download Cable User Guide*
- *MasterBlaster Serial/USB Communications Cable User Guide*
- *ByteBlaster II Download Cable User Guide*
- *ByteBlasterMV Download Cable User Guide*

Passive Parallel Asynchronous Configuration

Passive parallel asynchronous (PPA) configuration uses an intelligent host, such as a microprocessor, to transfer configuration data from a storage device, such as flash memory, to the target Arria GX device.

Configuration data can be stored in RBF, HEX, or TTF format. The host system outputs byte-wide data and the accompanying strobe signals to the device. When using PPA, pull the DCLK pin high through a 10-kΩ pull-up resistor to prevent unused configuration input pins from floating.

You cannot use the Arria GX decompression feature if you are configuring your Arria GX device using PPA mode.

Table 11–16 shows the MSEL pin settings when using the PS configuration scheme.

Table 11–16. Arria GX MSEL Pin Settings for PPA Configuration Schemes

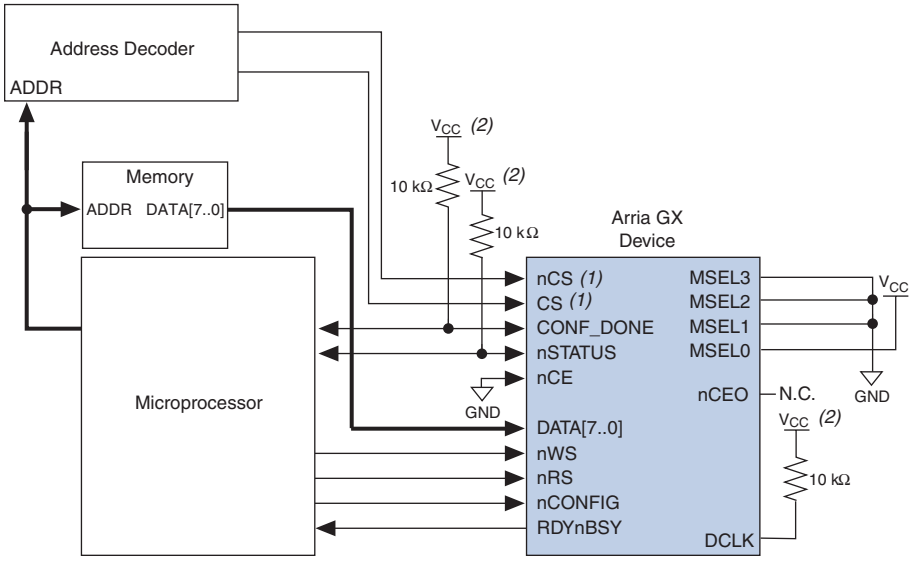
Configuration Scheme	MSEL3	MSEL2	MSEL1	MSEL0
PPA	0	0	0	1
Remote System Upgrade PPA *(1)*	0	1	0	1

Note to Table 11–16:

(1) This scheme requires that you drive the RUnLU pin to specify either remote update or local update. For more information about remote system upgrades in Arria GX devices, refer to the *Remote System Upgrades with Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*.

Figure 11–30 shows the configuration interface connections between the device and a microprocessor for single device PPA configuration. The microprocessor or an optional address decoder can control the device's chip select pins, nCS and CS. The address decoder allows the microprocessor to select the Arria GX device by accessing a particular address, which simplifies the configuration process. Hold the nCS and CS pins active during configuration and initialization.

Figure 11–30. Single Device PPA Configuration Using a Microprocessor



Notes to Figure 11–30:

(1) If not used, the CS pin can be connected to V_{CC} directly. If not used, the nCS pin can be connected to GND directly.

(2) The pull-up resistor should be connected to a supply that provides an acceptable input signal for the device. V_{CC} should be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

During PPA configuration, it is only required to use either the nCS or CS pin. Therefore, if you are using only one chip-select input, the other must be tied to the active state. For example, nCS can be tied to ground while CS is toggled to control configuration. The device's nCS or CS pins can be toggled during PPA configuration if the design meets the specifications set for t_{CSSU}, t_{WSP}, and t_{CSH} listed in Table 11–17 on page 11–80.

Upon power-up, the Arria GX devices go through a POR. The POR delay is dependent on the PORSEL pin setting. When PORSEL is driven low, the POR time is approximately 100 ms. If PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device will reset, hold nSTATUS low, and tri-state all user I/O pins. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *DC & Switching Characteristics* chapter in volume 1 of the *Arria GX Device Handbook*.

The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration, the microprocessor must generate a low-to-high transition on the nCONFIG pin.

To begin configuration, power the V_{CCINT}, V_{CCIO}, and V_{CCPD} voltages (for the banks where the configuration and JTAG pins reside) to the appropriate voltage levels.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. Once nSTATUS is released, the device is ready to receive configuration data and the configuration stage begins. When nSTATUS is pulled high, the microprocessor should then assert the target device's nCS pin low and/or CS pin high. Next, the microprocessor places an 8-bit configuration word (one byte) on the target device's DATA[7..0] pins and pulses the nWS pin low.

On the rising edge of nWS, the target device latches in a byte of configuration data and drives its RDYnBSY signal low, which indicates it is processing the byte of configuration data. The microprocessor can then perform other system functions while the Arria GX device is processing the byte of configuration data.

During the time RDYnBSY is low, the Arria GX device internally processes the configuration data using its internal oscillator (typically 100 MHz). When the device is ready for the next byte of configuration data, it will drive RDYnBSY high. If the microprocessor senses a high signal when it polls RDYnBSY, the microprocessor sends the next byte of configuration data to the device.

Alternatively, the nRS signal can be strobed low, causing the RDYnBSY signal to appear on DATA7. Because RDYnBSY does not need to be monitored, this pin doesn't need to be connected to the microprocessor. Do not drive data onto the data bus while nRS is low because it will cause contention on the DATA7 pin. If you are not using the nRS pin to monitor configuration, it should be tied high.

To simplify configuration and save an I/O port, the microprocessor can wait for the total time of t_{BUSY} (max) + t_{RDY2WS} + t_{W2SB} before sending the next data byte. In this set-up, nRS should be tied high and RDYnBSY does not need to be connected to the microprocessor. The t_{BUSY}, t_{RDY2WS}, and t_{W2SB} timing specifications are listed in Table 11–17 on page 11–80.

Next, the microprocessor checks nSTATUS and CONF_DONE. If nSTATUS is not low and CONF_DONE is not high, the microprocessor sends the next data byte. However, if nSTATUS is not low and all the configuration data has been received, the device is ready for initialization. The CONF_DONE pin will go high one byte early in parallel configuration (FPP and PPA) modes. The last byte is required for serial configuration (AS and PS) modes. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin. The open-drain CONF_DONE pin is pulled high by an external 10-kΩ pull-up resistor. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

In Arria GX devices, the initialization clock source is either the internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Arria GX device provides itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device.

You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. The **Enable user-supplied start-up clock** (**CLKUSR**) option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR does not affect the configuration process. After CONF_DONE goes high, CLKUSR is enabled after the time specified as t_{CD2CU}. After this time period elapses, Arria GX devices require 299 clock cycles to initialize properly and enter user mode. Arria GX devices support a CLKUSR f_{MAX} of 100 MHz.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. This **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used, it is high because of an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. The microprocessor must be able to detect this low-to-high transition that signals the device has entered user mode. When initialization is complete, the device enters user mode. In user-mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

To ensure DATA[7..0] is not left floating at the end of configuration, the microprocessor must drive them either high or low, whichever is convenient on your board. After configuration, the nCS, CS, nRS, nWS, RDYnBSY, and DATA[7..0] pins can be used as user I/O pins. When choosing the PPA scheme in the Quartus II software as a default, these I/O pins are tri-stated in user mode and should be driven by the microprocessor. To change this default option in the Quartus II software, select the **Dual-Purpose Pins** tab of the **Device & Pin Options** dialog box.

If an error occurs during configuration, the device drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the microprocessor that there is an error. If the **Auto-restart configuration after error** option-available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box-is turned on, the device releases nSTATUS after a reset time-out period (maximum of 100 µs). After nSTATUS is released and pulled high by a pull-up resistor, the microprocessor can try to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

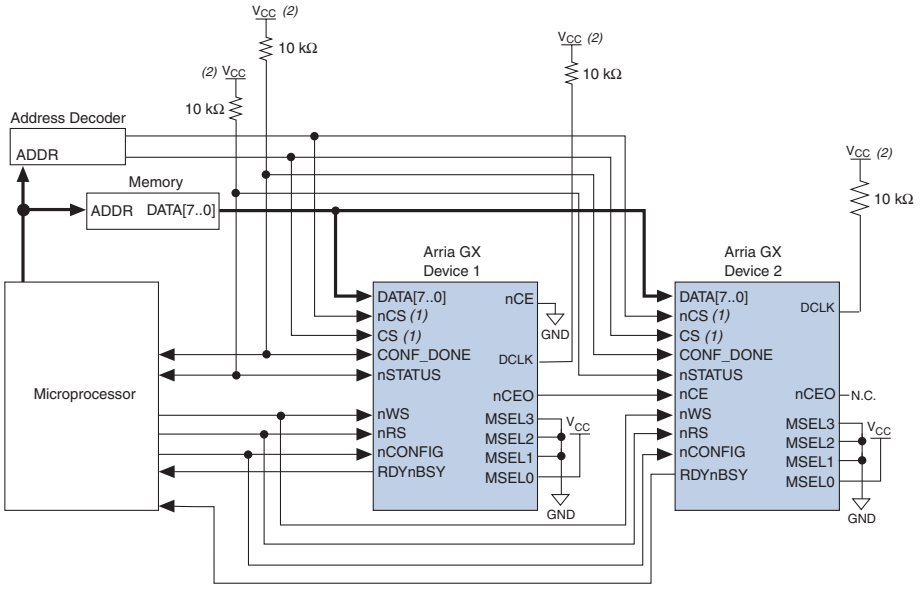
The microprocessor can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. To detect errors and determine when programming completes, monitor the CONF_DONE pin with the microprocessor. If the microprocessor sends all configuration data but CONF_DONE or INIT_DONE has not gone high, the microprocessor must reconfigure the target device.

If you are using the optional CLKUSR pin and nCONFIG is pulled low to restart configuration during device initialization, ensure CLKUSR continues toggling during the time nSTATUS is low (maximum of 100 µs).

When the device is in user-mode, a reconfiguration can be initiated by transitioning the nCONFIG pin low-to-high. The nCONFIG pin should go low for at least 2 µs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Once nCONFIG returns to a logic high level and nSTATUS is released by the device, reconfiguration begins.

Figure 11–31 shows how to configure multiple Arria GX devices using a microprocessor. This circuit is similar to the PPA configuration circuit for a single device, except the devices are cascaded for multi-device configuration.

Figure 11–31. Multi-Device PPA Configuration Using a Microprocessor



Notes to Figure 11–31:

(1) If not used, the CS pin can be connected to V_{CC} directly. If not used, the nCS pin can be connected to GND directly.

(2) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain. V_{CC} should be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

In the multi-device PPA configuration, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration within one clock cycle. Therefore, the transfer of data destinations is transparent to the microprocessor.

Each device's RDYnBSY pin can have a separate input to the microprocessor. Alternatively, if the microprocessor is pin limited, all the RDYnBSY pins can feed an AND gate and the output of the AND gate can feed the microprocessor. For example, if you have two devices in a PPA configuration chain, the second device's RDYnBSY pin will be high during the time that the first device is being configured. When the first device has been successfully configured, it will drive nCEO low to activate the next device in the chain and drive its RDYnBSY pin high. Therefore, because

RDYnBSY signal is driven high before configuration and after configuration before entering user-mode, the device being configured will govern the output of the AND gate.

The nRS signal can be used in the multi-device PPA chain because the Arria GX devices tri-state the DATA[7..0] pins before configuration and after configuration before entering user-mode to avoid contention. Therefore, only the device that is currently being configured responds to the nRS strobe by asserting DATA7.

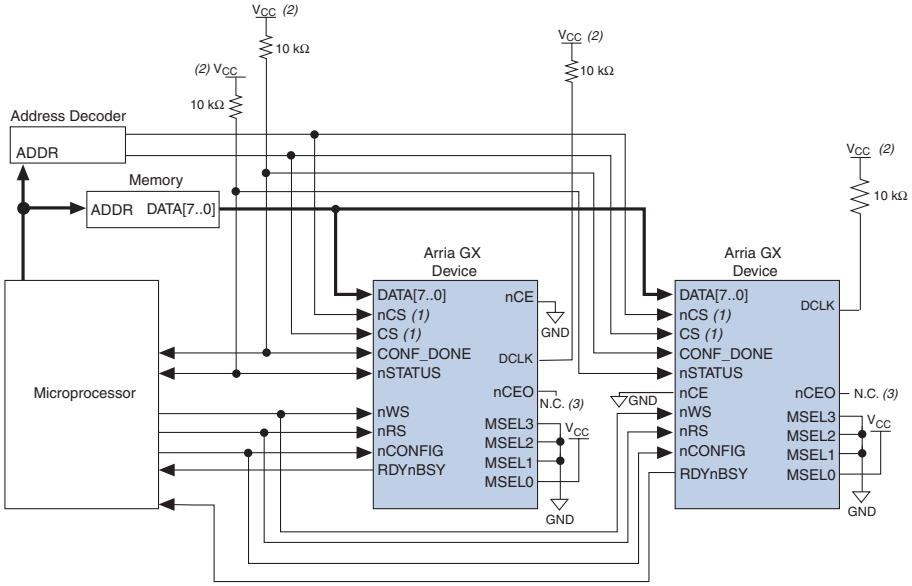
All other configuration pins (nCONFIG, nSTATUS, DATA[7..0], nCS, CS, nWS, nRS, and CONF_DONE) are connected to every device in the chain. It is not necessary to tie nCS and CS together for every device in the chain, as each device's nCS and CS input can be driven by a separate source. Configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Because all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

If the **Auto-restart configuration after error** option is turned on, the devices release their nSTATUS pins after a reset time-out period (maximum of 100 µs). After all nSTATUS pins are released and pulled high, the microprocessor can try to reconfigure the chain without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

In your system, you may have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DATA[7..0], nCS, CS, nWS, nRS, and CONF_DONE) are connected to every device in the chain. Configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices start and complete configuration at the same time. Figure 11–32 shows multi-device PPA configuration when both devices are receiving the same configuration data.

Figure 11–32. Multiple-Device PPA Configuration Using a Microprocessor When Both Devices Receive the Same Data



Notes to Figure 11–32:

(1) If not used, the CS pin can be connected to V_{CC} directly. If not used, the nCS pin can be connected to GND directly.

(2) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain. V_{CC} should be high enough to meet the V_{IH} specification of the I/O on the device and the external host.

(3) The nCEO pins of both devices are left unconnected when configuring the same configuration data into multiple devices.

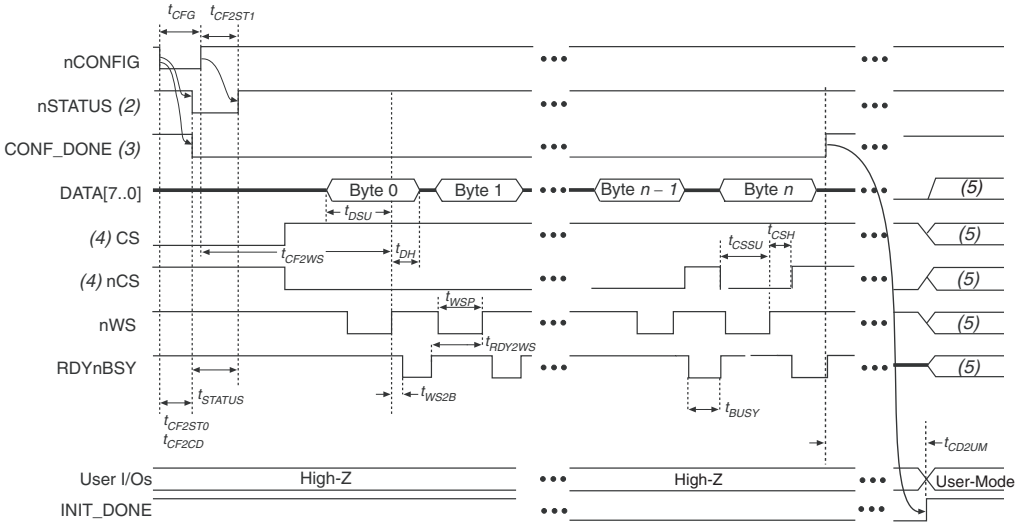
You can use a single configuration chain to configure Arria GX devices with other Altera devices that support PPA configuration, such as Stratix, Mercury™, APEX™ 20K, ACEX® 1K, and FLEX® 10KE devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information about configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook*.

PPA Configuration Timing

Figure 11–33 shows the timing waveform for the PPA configuration scheme using a microprocessor.

Figure 11–33. Arria GX PPA Configuration Timing Waveform Using nWS *Note (1)*



Notes to Figure 11–33:

(1) The beginning of this waveform shows the device in user-mode. In user-mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.

(2) Upon power-up, Arria GX devices hold nSTATUS low for the time of the POR delay.

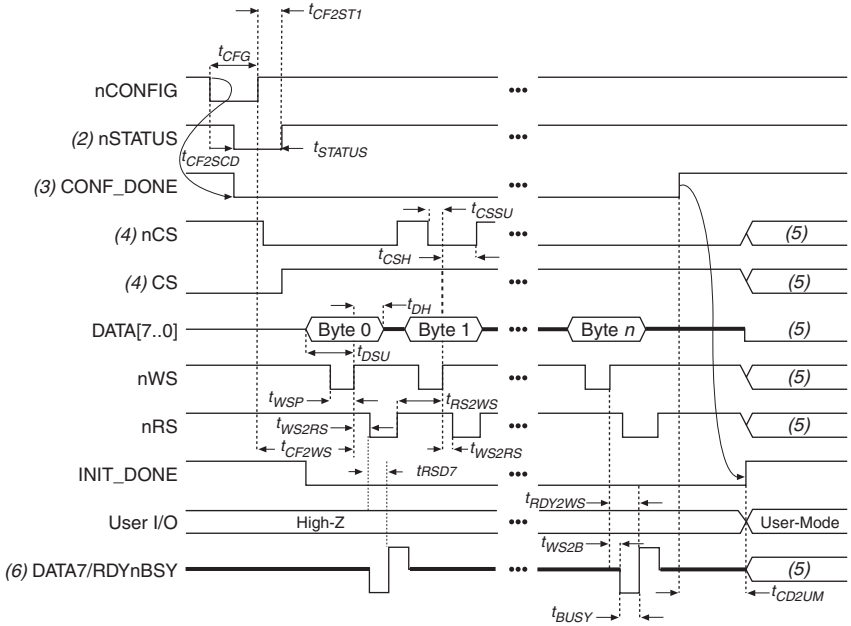
(3) Upon power-up, before and during configuration, CONF_DONE is low.

(4) The user can toggle nCS or CS during configuration if the design meets the specification for t_{CSSU}, t_{WSP}, and t_{CSH}.

(5) DATA[7..0], CS, nCS, nWS, nRS, and RDYnBSY are available as user I/O pins after configuration and the state of theses pins depends on the dual-purpose pin settings.

Figure 11–34 shows the timing waveform for the PPA configuration scheme when using a strobed nRS and nWS signal.

Figure 11–34. Arria GX PPA Configuration Timing Waveform Using nRS & nWS *Note (1)*



Notes to Figure 11–34:

(1) The beginning of this waveform shows the device in user-mode. In user-mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.
(2) Upon power-up, Arria GX devices hold nSTATUS low for the time of the POR delay.
(3) Upon power-up, before and during configuration, CONF_DONE is low.
(4) The user can toggle nCS or CS during configuration if the design meets the specification for t_{CSSU}, t_{WSP}, and t_{CSH}.
(5) DATA[7..0], CS, nCS, nWS, nRS, and RDYnBSY are available as user I/O pins after configuration and the state of theses pins depends on the dual-purpose pin settings.
(6) DATA7 is a bidirectional pin. It is an input for configuration data input, but it is an output to show the status of RDYnBSY.

Table 11–17 defines the timing parameters for Arria GX devices for PPA configuration.

Table 11–17. PPA Timing Parameters for Arria GX Devices (Part 1 of 2) *Note (1)*

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		800	ns

Table 11–17. PPA Timing Parameters for Arria GX Devices (Part 2 of 2) *Note (1)*

Symbol	Parameter	Min	Max	Units
t_{CFG}	nCONFIG low pulse width	2		µs
t_{STATUS}	nSTATUS low pulse width	10	100 *(2)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high		100 *(2)*	µs
t_{CSSU}	Chip select setup time before rising edge on nWS	10		ns
t_{CSH}	Chip select hold time after rising edge on nWS	0		ns
t_{CF2WS}	nCONFIG high to first rising edge on nWS	100		µs
t_{ST2WS}	nSTATUS high to first rising edge on nWS	2		µs
t_{DSU}	Data setup time before rising edge on nWS	10		ns
t_{DH}	Data hold time after rising edge on nWS	0		ns
t_{WSP}	nWS low pulse width	15		ns
t_{WS2B}	nWS rising edge to RDYnBSY low		20	ns
t_{BUSY}	RDYnBSY low pulse width	7	45	ns
t_{RDY2WS}	RDYnBSY rising edge to nWS rising edge	15		ns
t_{WS2RS}	nWS rising edge to nRS falling edge	15		ns
t_{RS2WS}	nRS rising edge to nWS rising edge	15		ns
t_{RSD7}	nRS falling edge to DATA7 valid with RDYnBSY signal		20	ns
t_R	Input rise time		40	ns
t_F	Input fall time		40	ns
t_{CD2UM}	CONF_DONE high to user mode *(3)*	20	100	µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	40		ns
t_{CD2UMC}	CONF_DONE high to user mode with **CLKUSR** option on	t_{CD2CU} + (299 × CLKUSR period)		

Notes to Table 11–17:

(1) This information is preliminary.

(2) This value is obtainable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(3) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device.

Device configuration options and how to create configuration files are discussed further in the *Software Settings* section of the *Configuration Handbook*.

JTAG Configuration

The JTAG has developed a specification for boundary-scan testing (BST). This boundary-scan test architecture offers the capability to efficiently test components on PCBs with tight lead spacing. The BST architecture can test pin connections without using physical test probes and capture functional data while a device is operating normally. The JTAG circuitry can also be used to shift configuration data into the device. The Quartus II software automatically generates SOFs that can be used for JTAG configuration with a download cable in the Quartus II software programmer.

For more information on JTAG boundary-scan testing, refer to the following documents:

- *IEEE 1149.1 (JTAG) Boundary-Scan Testing for Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*
- *Jam Programming Support - JTAG Technologies*

Arria GX devices are designed such that JTAG instructions have precedence over any device configuration modes. Therefore, JTAG configuration can take place without waiting for other configuration modes to complete. For example, if you attempt JTAG configuration of Arria GX devices during PS configuration, PS configuration is terminated and JTAG configuration begins.

You cannot use the Arria GX decompression feature if you are configuring your Arria GX device when using JTAG-based configuration.

A device operating in JTAG mode uses four required pins, `TDI`, `TDO`, `TMS`, and `TCK`, and one optional pin, `TRST`. The `TCK` pin has an internal weak pull-down resistor, while the `TDI`, `TMS`, and `TRST` pins have weak internal pull-up resistors (typically 25 kΩ). The `TDO` output pin is powered by V_{CCIO} in I/O bank 4. All of the JTAG input pins are powered by the 3.3-V V_{CCPD} pin. All user I/O pins are tri-stated during JTAG configuration. Table 11–18 explains each JTAG pin's function.

The `TDO` output is powered by the V_{CCIO} power supply of I/O bank 4.

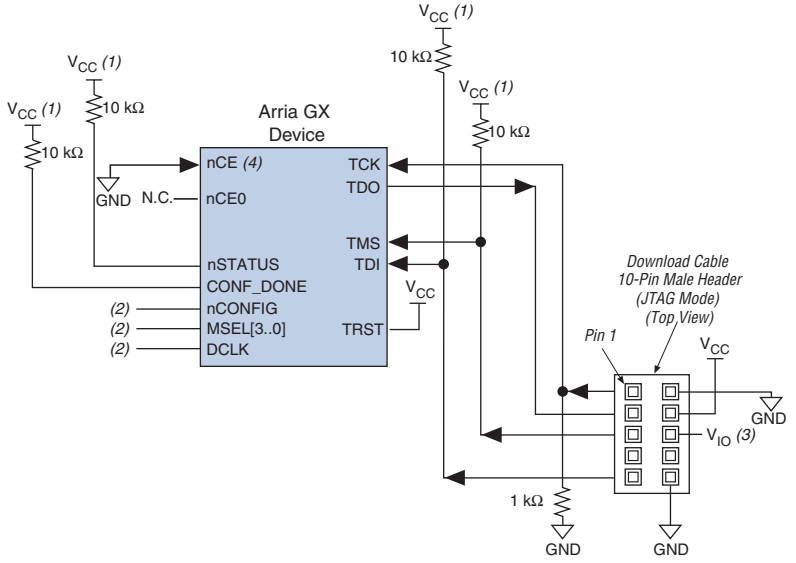
For recommendations on how to connect a JTAG chain with multiple voltages across the devices in the chain, refer to the *IEEE 1149.1 (JTAG) Boundary-Scan Testing for Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*.

Table 11–18. Dedicated JTAG Pins

Pin Name	Pin Type	Description
TDI	Test data input	Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}.
TDO	Test data output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by leaving this pin unconnected.
TMS	Test mode select	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}.
TCK	Test clock input	The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND.
TRST	Test reset input (optional)	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to IEEE Std. 1149.1. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND.

During JTAG configuration, data can be downloaded to the device on the PCB through the USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV download cable. Configuring devices through a cable is similar to programming devices in-system, except the TRST pin should be connected to V_{CC}. This ensures that the TAP controller is not reset. Figure 11–35 shows JTAG configuration of a single Arria GX device.

Figure 11–35. JTAG Configuration of a Single Device Using a Download Cable



Notes to Figure 11–35:

(1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster (V_{IO} pin), ByteBlaster II, or ByteBlasterMV cable.

(2) The nCONFIG, MSEL[3..0] pins should be connected to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect nCONFIG to V_{CC}, and MSEL[3..0] to ground. Pull DCLK either high or low, whichever is convenient on your board.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. Refer to the *MasterBlaster Serial/USB Communications Cable User Guide* for this value. In the ByteBlasterMV cable, this pin is a no connect. In the USB Blaster and ByteBlaster II cables, this pin is connected to nCE when it is used for active serial programming, otherwise it is a no connect.

(4) nCE must be connected to GND or driven low for successful JTAG configuration.

To configure a single device in a JTAG chain, the programming software places all other devices in bypass mode. In bypass mode, devices pass programming data from the TDI pin to the TDO pin through a single bypass register without being affected internally. This scheme enables the programming software to program or verify the target device. Configuration data driven into the device appears on the TDO pin one clock cycle later.

The Quartus II software verifies successful JTAG configuration upon completion. At the end of configuration, the software checks the state of CONF_DONE through the JTAG port. When Quartus II generates a (**.jam**) file for a multi-device chain, it contains instructions so that all the devices in the chain will be initialized at the same time. If CONF_DONE is not high, the Quartus II software indicates that configuration has failed. If

CONF_DONE is high, the software indicates that configuration was successful. After the configuration bitstream is transmitted serially via the JTAG TDI port, the TCK port is clocked an additional 299 cycles to perform device initialization.

Arria GX devices have dedicated JTAG pins that always function as JTAG pins. Not only can you perform JTAG testing on Arria GX devices before and after, but also during configuration. While other device families do not support JTAG testing during configuration, Arria GX devices support the bypass, idcode, and sample instructions during configuration without interrupting configuration. All other JTAG instructions may only be issued by first interrupting configuration and reprogramming I/O pins using the CONFIG_IO instruction.

The CONFIG_IO instruction allows I/O buffers to be configured via the JTAG port and when issued, interrupts configuration. This instruction allows you to perform board-level testing prior to configuring the Arria GX device or waiting for a configuration device to complete configuration. Once configuration has been interrupted and JTAG testing is complete, the part must be reconfigured via JTAG (PULSE_CONFIG instruction) or by pulsing nCONFIG low.

The chip-wide reset (DEV_CLRn) and chip-wide output enable (DEV_OE) pins on Arria GX devices do not affect JTAG boundary-scan or programming operations. Toggling these pins does not affect JTAG operations (other than the usual boundary-scan operation).

When designing a board for JTAG configuration of Arria GX devices, consider the dedicated configuration pins. Table 11–19 shows how these pins should be connected during JTAG configuration.

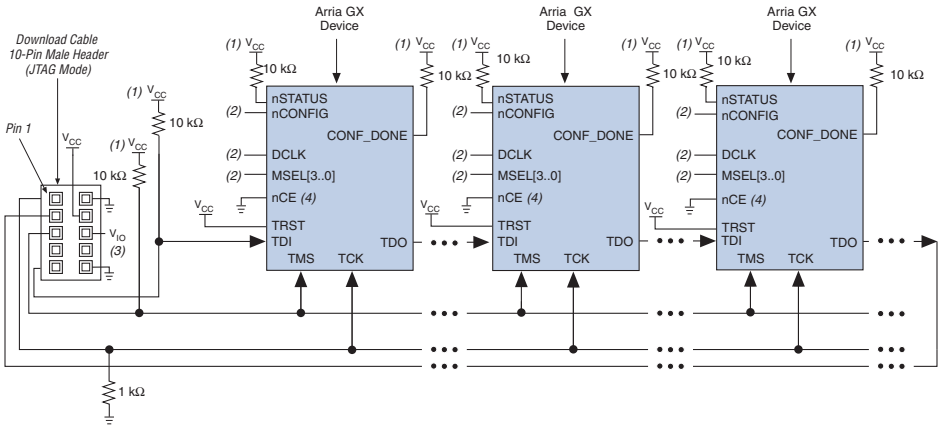
When programming a JTAG device chain, one JTAG-compatible header is connected to several devices. The number of devices in the JTAG chain is limited only by the drive capability of the download cable. When four or more devices are connected in a JTAG chain, Altera recommends buffering the TCK, TDI, and TMS pins with an on-board buffer.

Table 11–19. Dedicated Configuration Pin Connections During JTAG Configuration

Signal	Description
nCE	On all Arria GX devices in the chain, nCE should be driven low by connecting it to ground, pulling it low via a resistor, or driving it by some control circuitry. For devices that are also in multi-device FPP, AS, PS, or PPA configuration chains, the nCE pins should be connected to GND during JTAG configuration or JTAG configured in the same order as the configuration chain.
nCEO	On all Arria GX devices in the chain, nCEO can be left floating or connected to the nCE of the next device.
MSEL	These pins must not be left floating. These pins support whichever non-JTAG configuration is used in production. If only JTAG configuration is used, tie these pins to ground.
nCONFIG	Driven high by connecting to V_{CC}, pulling up via a resistor, or driven high by some control circuitry.
nSTATUS	Pull to V_{CC} via a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each nSTATUS pin should be pulled up to V_{CC} individually.
CONF_DONE	Pull to V_{CC} via a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each CONF_DONE pin should be pulled up to V_{CC} individually. CONF_DONE going high at the end of JTAG configuration indicates successful configuration.
DCLK	Should not be left floating. Drive low or high, whichever is more convenient on your board.

JTAG-chain device programming is ideal when the system contains multiple devices, or when testing your system using JTAG BST circuitry. Figure 11–36 shows multi-device JTAG configuration.

Figure 11–36. JTAG Configuration of Multiple Devices Using a Download Cable



Notes to Figure 11–36:

(1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster (V_{IO} pin), ByteBlaster II, or ByteBlasterMV cable.

(2) The nCONFIG and MSEL[3..0] pins should be connected to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect nCONFIG to V_{CC}, and MSEL[3..0] to ground. Pull DCLK either high or low, whichever is convenient on your board.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. Refer to the *MasterBlaster Serial/USB Communications Cable User Guide* for this value. In the ByteBlasterMV cable, this pin is a no connect. In the USB Blaster and ByteBlaster II cables, this pin is connected to nCE when it is used for active serial programming, otherwise it is a no connect.

(4) nCE must be connected to GND or driven low for successful JTAG configuration.

The nCE pin must be connected to GND or driven low during JTAG configuration. In multi-device FPP, AS, PS, and PPA configuration chains, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. In addition, the CONF_DONE and nSTATUS signals are all shared in multi-device FPP, AS, PS, or PPA configuration chains so the devices can enter user mode at the same time after configuration is complete. When the CONF_DONE and nSTATUS signals are shared among all the devices, every device must be configured when JTAG configuration is performed.

If you only use JTAG configuration, Altera recommends that you connect the circuitry as shown in Figure 11–36, where each of the CONF_DONE and nSTATUS signals are isolated, so that each device can enter user mode individually.

After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. Therefore, if these devices are also in a JTAG chain, make sure the nCE pins are connected to GND during JTAG configuration or that the devices are JTAG configured in the same order as the configuration chain. As long as the devices are JTAG configured in the same order as the multi-device configuration chain, the nCEO of the previous device will drive nCE of the next device low when it has successfully been JTAG configured.

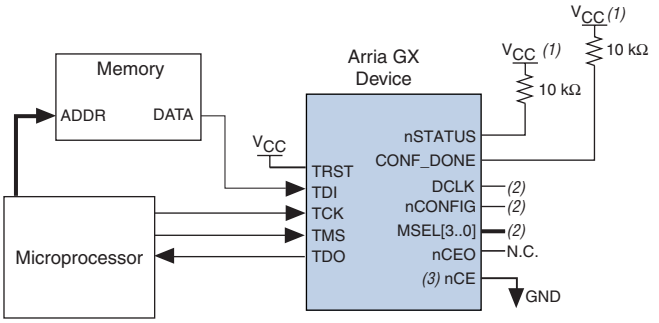
Other Altera devices that have JTAG support can be placed in the same JTAG chain for device programming and configuration.

Stratix, Arria GX, Cyclone®, and Cyclone II devices must be within the first 17 devices in a JTAG chain. All of these devices have the same JTAG controller. If any of the Stratix, Arria GX, Cyclone, and Cyclone II devices are in the 18th or after they will fail configuration. This does not affect SignalTap® II.

For more information on configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook*.

Figure 11–37 shows JTAG configuration of a Arria GX device with a microprocessor.

Figure 11–37. JTAG Configuration of a Single Device Using a Microprocessor



Notes to Figure 11–37:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain. V_{CC} should be high enough to meet the V_{IH} specification of the I/O on the device.

(2) The `nCONFIG`, `MSEL[3..0]` pins should be connected to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect `nCONFIG` to V_{CC}, and `MSEL[3..0]` to ground. Pull `DCLK` either high or low, whichever is convenient on your board.

(3) `nCE` must be connected to GND or driven low for successful JTAG configuration.

Jam STAPL

Jam STAPL, JEDEC standard JESD-71, is a standard file format for in-system programmability (ISP) purposes. Jam STAPL supports programming or configuration of programmable devices and testing of electronic systems, using the IEEE 1149.1 JTAG interface. Jam STAPL is a freely licensed open standard.

The Jam Player provides an interface for manipulating the IEEE Std. 1149.1 JTAG TAP state machine.

For more information on JTAG and Jam STAPL in embedded environments, refer to *AN 122: Using Jam STAPL for ISP & ICR via an Embedded Processor*.

Device Configuration Pins

Table 11–20 describes the connections and functionality of all the configuration related pins on Arria GX devices and summarizes the Arria GX pin configuration.

Table 11–20. Arria GX Configuration Pin Summary (Part 1 of 2) *Note (1)*

Bank	Description	Input/Output	Dedicated	Powered By	Configuration Mode
3	PGM[2..0]	Output		*(2)*	PS, FPP, PPA, RU, LU
3	ASDO	Output		*(2)*	AS
3	nCSO	Output		*(2)*	AS
3	CRC_ERROR	Output		*(2)*	Optional, all modes
3	DATA0	Input		*(3)*	All modes except JTAG
3	DATA[7..1]	Input		*(3)*	FPP, PPA
3	DATA7	Bidirectional		*(2)*, *(3)*	PPA
3	RDYnBSY	Output		*(2)*	PPA
3	INIT_DONE	Output		Pull-up	Optional, all modes
3	nSTATUS	Bidirectional	Yes	Pull-up	All modes
3	nCE	Input	Yes	*(3)*	All modes
3	DCLK	Input	Yes	*(3)*	PS, FPP
		Output		*(2)*	AS
3	CONF_DONE	Bidirectional	Yes	Pull-up	All modes
8	TDI	Input	Yes	VCCPD	JTAG
8	TMS	Input	Yes	VCCPD	JTAG
8	TCK	Input	Yes	VCCPD	JTAG
8	TRST	Input	Yes	VCCPD	JTAG
8	nCONFIG	Input	Yes	*(3)*	All modes
8	VCCSEL	Input	Yes	VCCINT	All modes
8	CS	Input		*(3)*	PPA
8	CLKUSR	Input		*(3)*	Optional
8	nWS	Input		*(3)*	PPA
8	nRS	Input		*(3)*	PPA
8	RUnLU	Input		*(3)*	PS, FPP, PPA, RU, LU
8	nCS	Input		*(3)*	PPA
7	PORSEL	Input	Yes	VCCINT	All modes
7	nIO_PULLUP	Input	Yes	VCCINT	All modes
7	PLL_ENA	Input	Yes	*(3)*	Optional

Table 11–20. Arria GX Configuration Pin Summary (Part 2 of 2) Note (1)					
Bank	**Description**	**Input/Output**	**Dedicated**	**Powered By**	**Configuration Mode**
7	nCEO	Output	Yes	(2), (4)	All modes
4	MSEL[3..0]	Input	Yes	VCCINT	All modes
4	TDO	Output	Yes	(2), (4)	JTAG

Notes to Table 11–20:

(1) Total number of pins is 41, total number of dedicated pins is 19.

(2) All outputs are powered by VCCIO except as noted.

(3) All inputs are powered by VCCIO or VCCPD, based on the VCCSEL setting, except as noted.

(4) An external pull-up resistor may be required for this configuration pin because of the multivolt I/O interface. Refer to the *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook* for pull-up or level shifter recommendations for nCEO and TDO.

Figure 11–38 shows the I/O bank locations.

Figure 11–38. Arria GX I/O Bank Numbers

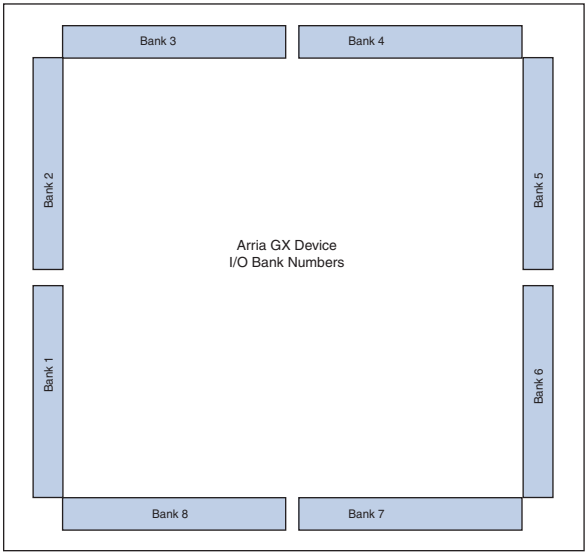


Table 11–21 describes the dedicated configuration pins, which are required to be connected properly on your board for successful configuration. Some of these pins may not be required for your configuration schemes.

Table 11–21. Dedicated Configuration Pins on the Arria GX Device (Part 1 of 10)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
V_{CCPD}	N/A	All	Power	Dedicated power pin. This pin is used to power the I/O pre-drivers, the JTAG input pins, and the configuration input pins that are affected by the voltage level of VCCSEL. This pin must be connected to 3.3-V. V_{CCPD} must ramp-up from 0-V to 3.3-V within 100 ms. If V_{CCPD} is not ramped up within this specified time, your Arria GX device will not configure successfully. If your system does not allow for a V_{CCPD} ramp-up time of 100 ms or less, you must hold nCONFIG low until all power supplies are stable.
VCCSEL	N/A	All	Input	Dedicated input that selects which input buffer is used on the PLL_ENA pin and the configuration input pins; nCONFIG, DCLK (when used as an input), nSTATUS (when used as an input), CONF_DONE (when used as an input), DEV_OE, DEV_CLRn, DATA[7..0], RUnLU, nCE, nWS, nRS, CS, nCS, and CLKUSR. The 3.3-V/2.5-V input buffer is powered by V_{CCPD}, while the 1.8-V/1.5-V input buffer is powered by V_{CCIO}. The VCCSEL input buffer has an internal 5-kΩ pull-down resistor that is always active. The VCCSEL input buffer is powered by V_{CCINT} and must be hardwired to V_{CCPD} or ground. A logic high selects the 1.8-V/1.5-V input buffer, and a logic low selects the 3.3-V/2.5-V input buffer. For more information, refer to the "VCCSEL Pin" section.

Table 11–21. Dedicated Configuration Pins on the Arria GX Device (Part 2 of 10)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
PORSEL	N/A	All	Input	Dedicated input which selects between a POR time of 12 ms or 100 ms. A logic high (1.5 V, 1.8 V, 2.5 V, 3.3 V) selects a POR time of about 12 ms and a logic low selects POR time of about 100 ms. The PORSEL input buffer is powered by V_{CCINT} and has an internal 5-kΩ pull-down resistor that is always active. The PORSEL pin should be tied directly to V_{CCPD} or GND.
nIO_PULLUP	N/A	All	Input	Dedicated input that chooses whether the internal pull-up resistors on the user I/O pins and dual-purpose I/O pins (nCSO, nASDO, DATA[7..0], nWS, nRS, RDYnBSY, nCS, CS, RUnLU, PGM[], CLKUSR, INIT_DONE, DEV_OE, DEV_CLR) are on or off before and during configuration. A logic high (1.5 V, 1.8 V, 2.5 V, 3.3 V) turns off the weak internal pull-up resistors, while a logic low turns them on. The nIO-PULLUP input buffer is powered by V_{CCPD} and has an internal 5-kΩ pull-down resistor that is always active. The nIO-PULLUP can be tied directly to V_{CCPD} or use a 1-kΩ pull-up resistor or tied directly to GND.
MSEL[3..0]	N/A	All	Input	4-bit configuration input that sets the Arria GX device configuration scheme. Refer to Table 11–1 for the appropriate connections. These pins must be hard-wired to V_{CCPD} or GND. The MSEL[3..0] pins have internal 5-kΩ pull-down resistors that are always active.

Table 11–21. Dedicated Configuration Pins on the Arria GX Device (Part 3 of 10)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nCONFIG	N/A	All	Input	Configuration control input. Pulling this pin low during user-mode will cause the device to lose its configuration data, enter a reset state, tri-state all I/O pins. Returning this pin to a logic high level will initiate a reconfiguration. If your configuration scheme uses an enhanced configuration device or EPC2 device, nCONFIG can be tied directly to V_{CC} or to the configuration device's nINIT_CONF pin.
nSTATUS	N/A	All	Bidirectional open-drain	The device drives nSTATUS low immediately after power-up and releases it after the POR time. Status output. If an error occurs during configuration, nSTATUS is pulled low by the target device. Status input. If an external source drives the nSTATUS pin low during configuration or initialization, the target device enters an error state. Driving nSTATUS low after configuration and initialization does not affect the configured device. If a configuration device is used, driving nSTATUS low will cause the configuration device to attempt to configure the device, but because the device ignores transitions on nSTATUS in user-mode, the device does not reconfigure. To initiate a reconfiguration, nCONFIG must be pulled low. The enhanced configuration devices' and EPC2 devices' OE and nCS pins have optional internal programmable pull-up resistors. If internal pull-up resistors on the enhanced configuration device are used, external 10-kΩ pull-up resistors should not be used on these pins. The external 10-kΩ pull-up resistor should be used only when the internal pull-up resistor is not used.

Table 11–21. Dedicated Configuration Pins on the Arria GX Device (Part 4 of 10)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nSTATUS (continued)				If VCCPD and VCCIO are not fully powered up, the following could occur: • VCCPD and VCCIO are powered high enough for the nSTATUS buffer to function properly, and nSTATUS is driven low. When VCCPD and VCCIO are ramped up, POR trips, and nSTATUS is released after POR expires. • VCCPD and VCCIO are not powered high enough for the nSTATUS buffer to function properly. In this situation, nSTATUS might appear logic high, triggering a configuration attempt that would fail because POR did not yet trip. When VCCPD and VCCIO are powered up, nSTATUS is pulled low because POR did not yet trip. When POR trips after VCCPD and VCCIO are powered up, nSTATUS is released and pulled high. At that point, reconfiguration is triggered and the device is configured.

Table 11–21. Dedicated Configuration Pins on the Arria GX Device (Part 5 of 10)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
CONF_DONE	N/A	All	Bidirectional open-drain	Status output. The target device drives the CONF_DONE pin low before and during configuration. Once all configuration data is received without error and the initialization cycle starts, the target device releases CONF_DONE. Status input. After all data is received and CONF_DONE goes high, the target device initializes and enters user mode. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize. Driving CONF_DONE low after configuration and initialization does not affect the configured device. The enhanced configuration devices' and EPC2 devices' OE and nCS pins have optional internal programmable pull-up resistors. If internal pull-up resistors on the enhanced configuration device are used, external 10-kΩ pull-up resistors should not be used on these pins. The external 10-kΩ pull-up resistor should be used only when the internal pull-up resistor is not used.
nCE	N/A	All	Input	Active-low chip enable. The nCE pin activates the device with a low signal to allow configuration. The nCE pin must be held low during configuration, initialization, and user mode. In single device configuration, it should be tied low. In multi-device configuration, nCE of the first device is tied low while its nCEO pin is connected to nCE of the next device in the chain. The nCE pin must also be held low for successful JTAG programming of the device.

Table 11–21. Dedicated Configuration Pins on the Arria GX Device (Part 6 of 10)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nCEO	N/A	All	Output	Output that drives low when device configuration is complete. In single device configuration, this pin is left floating. In multi-device configuration, this pin feeds the next device's nCE pin. The nCEO of the last device in the chain is left floating. The nCEO pin is powered by V_{CCIO} in I/O bank 7. For recommendations on how to connect nCEO in a chain with multiple voltages across the devices in the chain, refer to the *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook*.
ASDO	N/A in AS mode I/O in non-AS mode	AS	Output	Control signal from the Arria GX device to the serial configuration device in AS mode used to read out configuration data. In AS mode, ASDO has an internal pull-up resistor that is always active.
nCSO	N/A in AS mode I/O in non-AS mode	AS	Output	Output control signal from the Arria GX device to the serial configuration device in AS mode that enables the configuration device. In AS mode, nCSO has an internal pull-up resistor that is always active.

Table 11–21. Dedicated Configuration Pins on the Arria GX Device (Part 7 of 10)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
DCLK	N/A	Synchronous configuration schemes (PS, FPP, AS)	Input (PS, FPP) Output (AS)	In PS and FPP configuration, DCLK is the clock input used to clock data from an external source into the target device. Data is latched into the device on the rising edge of DCLK. In AS mode, DCLK is an output from the Arria GX device that provides timing for the configuration interface. In AS mode, DCLK has an internal pull-up resistor (typically 25 kΩ) that is always active. In PPA mode, DCLK should be tied high to V_{CC} to prevent this pin from floating. After configuration, this pin is tri-stated. In schemes that use a configuration device, DCLK will be driven low after configuration is done. In schemes that use a control host, DCLK should be driven either high or low, whichever is more convenient. Toggling this pin after configuration does not affect the configured device.
DATA0	I/O	PS, FPP, PPA, AS	Input	Data input. In serial configuration modes, bit-wide configuration data is presented to the target device on the DATA0 pin. The V_{IH} and V_{IL} levels for this pin are dependent on the input buffer selected by the VCCSEL pin. Refer to the section "VCCSEL Pin" on page 11–9 for more information. In AS mode, DATA0 has an internal pull-up resistor that is always active. After configuration, DATA0 is available as a user I/O pin and the state of this pin depends on the **Dual-Purpose Pin** settings. After configuration, EPC1 and EPC1441 devices tri-state this pin, while enhanced configuration and EPC2 devices drive this pin high.

Table 11–21. Dedicated Configuration Pins on the Arria GX Device (Part 8 of 10)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
DATA[7..1]	I/O	Parallel configuration schemes (FPP and PPA)	Inputs	Data inputs. Byte-wide configuration data is presented to the target device on DATA[7..0]. The V_{IH} and V_{IL} levels for this pin are dependent on the input buffer selected by the VCCSEL pin. Refer to the section "VCCSEL Pin" on page 11–9 for more information. In serial configuration schemes, they function as user I/O pins during configuration, which means they are tri-stated. After PPA or FPP configuration, DATA[7..1] are available as user I/O pins and the state of these pin depends on the **Dual-Purpose Pin** settings.
DATA7	I/O	PPA	Bidirectional	In the PPA configuration scheme, the DATA7 pin presents the RDYnBSY signal after the nRS signal has been strobed low. The V_{IH} and V_{IL} levels for this pin are dependent on the input buffer selected by the VCCSEL pin. Refer to the section "VCCSEL Pin" on page 11–9 for more information. In serial configuration schemes, it functions as a user I/O pin during configuration, which means it is tri-stated. After PPA configuration, DATA7 is available as a user I/O and the state of this pin depends on the **Dual-Purpose Pin** settings.
nWS	I/O	PPA	Input	Write strobe input. A low-to-high transition causes the device to latch a byte of data on the DATA[7..0] pins. In non-PPA schemes, it functions as a user I/O pin during configuration, which means it is tri-stated. After PPA configuration, nWS is available as a user I/O pins and the state of this pin depends on the **Dual-Purpose Pin** settings.

Table 11–21. Dedicated Configuration Pins on the Arria GX Device (Part 9 of 10)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nRS	I/O	PPA	Input	Read strobe input. A low input directs the device to drive the RDYnBSY signal on the DATA7 pin. If the nRS pin is not used in PPA mode, it should be tied high. In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, nRS is available as a user I/O pin and the state of this pin depends on the **Dual-Purpose Pin** settings.
RDYnBSY	I/O	PPA	Output	Ready output. A high output indicates that the target device is ready to accept another data byte. A low output indicates that the target device is busy and not ready to receive another data byte. In PPA configuration schemes, this pin will drive out high after power-up, before configuration and after configuration before entering user-mode. In non-PPA schemes, it functions as a user I/O pin during configuration, which means it is tri-stated. After PPA configuration, RDYnBSY is available as a user I/O pin and the state of this pin depends on the **Dual-Purpose Pin** settings.

Table 11–21. Dedicated Configuration Pins on the Arria GX Device (Part 10 of 10)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nCS/CS	I/O	PPA	Input	Chip-select inputs. A low on nCS and a high on CS select the target device for configuration. The nCS and CS pins must be held active during configuration and initialization. During the PPA configuration mode, it is only required to use either the nCS or CS pin. Therefore, if only one chip-select input is used, the other must be tied to the active state. For example, nCS can be tied to ground while CS is toggled to control configuration. In non-PPA schemes, it functions as a user I/O pin during configuration, which means it is tri-stated. After PPA configuration, nCS and CS are available as user I/O pins and the state of these pins depends on the **Dual-Purpose Pin** settings.
RUnLU	N/A if using Remote System Upgrade I/O if not	Remote System Upgrade in FPP, PS, or PPA	Input	Input that selects between remote update and local update. A logic high (1.5-V, 1.8-V, 2.5-V, 3.3-V) selects remote update and a logic low selects local update. When not using remote update or local update configuration modes, this pin is available as general-purpose user I/O pin. When using remote system upgrade in AS mode, set the RUnLU pin to high because AS does not support local update
PGM[2..0]	N/A if using Remote System Upgrade I/O if not using	Remote System Upgrade in FPP, PS, or PPA	Output	These output pins select one of eight pages in the memory (either flash or enhanced configuration device) when using a remote system upgrade mode. When not using remote update or local update configuration modes, these pins are available as general-purpose user I/O pins.

Table 11–22 describes the optional configuration pins. If these optional configuration pins are not enabled in the Quartus II software, they are available as general-purpose user I/O pins. Therefore, during configuration, these pins function as user I/O pins and are tri-stated with weak pull-up resistors.

Table 11–22. Optional Configuration Pins

Pin Name	User Mode	Pin Type	Description
CLKUSR	N/A if option is on. I/O if option is off.	Input	Optional user-supplied clock input synchronizes the initialization of one or more devices. This pin is enabled by turning on the **Enable user-supplied start-up clock** (**CLKUSR**) option in the Quartus II software.
INIT_DONE	N/A if option is on. I/O if option is off.	Output open-drain	Status pin can be used to indicate when the device has initialized and is in user mode. When nCONFIG is low and during the beginning of configuration, the INIT_DONE pin is tri-stated and pulled high due to an external 10-kΩ pull-up resistor. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low. When initialization is complete, the INIT_DONE pin will be released and pulled high and the device enters user mode. Thus, the monitoring circuitry must be able to detect a low-to-high transition. This pin is enabled by turning on the **Enable INIT_DONE output** option in the Quartus II software.
DEV_OE	N/A if option is on. I/O if option is off.	Input	Optional pin that allows the user to override all tri-states on the device. When this pin is driven low, all I/O pins are tri-stated; when this pin is driven high, all I/O pins behave as programmed. This pin is enabled by turning on the **Enable device-wide output enable** (**DEV_OE**) option in the Quartus II software.
DEV_CLRn	N/A if option is on. I/O if option is off.	Input	Optional pin that allows you to override all clears on all device registers. When this pin is driven low, all registers are cleared; when this pin is driven high, all registers behave as programmed. This pin is enabled by turning on the **Enable device-wide reset** (**DEV_CLRn**) option in the Quartus II software.

Table 11–23 describes the dedicated JTAG pins. JTAG pins must be kept stable before and during configuration to prevent accidental loading of JTAG instructions. The TDI, TMS, and TRST have weak internal pull-up resistors (typically 25 kΩ) while TCK has a weak internal pull-down resistor. If you plan to use the SignalTap embedded logic array analyzer, you need to connect the JTAG pins of the Arria GX device to a JTAG header on your board.

Table 11–23. Dedicated JTAG Pins

Pin Name	User Mode	Pin Type	Description
TDI	N/A	Input	Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. The TDI pin is powered by the 3.3-V V_{CCPD} supply. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}.
TDO	N/A	Output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. The TDO pin is powered by V_{CCIO} in I/O bank 4. For recommendations on connecting a JTAG chain with multiple voltages across the devices in the chain, refer to the chapter *IEEE 1149.1 (JTAG) Boundary Scan Testing in Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by leaving this pin unconnected.
TMS	N/A	Input	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. The TMS pin is powered by the 3.3-V V_{CCPD} supply. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}.
TCK	N/A	Input	The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. The TCK pin is powered by the 3.3-V V_{CCPD} supply. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting TCK to GND.
TRST	N/A	Input	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to IEEE Std. 1149.1. The TRST pin is powered by the 3.3-V V_{CCPD} supply. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting the TRST pin to GND.

Conclusion

Arria GX devices can be configured in a number of different schemes to fit your system's need. In addition, configuration data decompression and remote system upgrade support supplement the Arria GX configuration solution.

Referenced Documents

This chapter references the following documents:

- *Altera Enhanced Configuration Devices* chapter in volume 2 of the *Configuration Handbook*
- *AN 122: Using Jam STAPL for ISP & ICR via an Embedded Processor*
- *AN 418: SRunner: An Embedded Solution for Serial Configuration*
- *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook*
- *Arria GX Device Handbook*
- *ByteBlaster II Download Cable User Guide*
- *ByteBlasterMV Download Cable User Guide*
- *Configuration Devices for SRAM-Based LUT Devices Data Sheet* chapter in volume 2 of the *Configuration Handbook*
- *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook*
- *DC & Switching Characteristics* chapter in volume 1 of the *Arria GX Device Handbook*
- *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* chapter in volume 2 of the *Configuration Handbook*
- *IEEE 1149.1 (JTAG) Boundary-Scan Testing for Arria GX Devices*
- Jam Programming Support - JTAG Technologies
- *MasterBlaster Serial/USB Communications Cable User Guide*
- *Remote System Upgrades With Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*
- *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 2 of the *Configuration Handbook*
- *Software Settings* section in volume 2 of the *Configuration Handbook*
- *USB-Blaster USB Port Download Cable User Guide*

Document Revision History

Table 11–24 shows the revision history for this chapter.

Table 11–24. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
May 2008 v1.3	Updated: • Table 11–2 • Figure 11–6 • Figure 11–7	—
	Minor text edits.	—
August 2007 v1.2	Added the “Referenced Documents” section.	—
	Minor text edits.	—
June 2007 v1.1	Updated t_{CF2CK} in Figures 11–6 and 11–7.	—
May 2007 v1.0	Initial Release	—

1. Configuring Altera FPGAs

CF51001-2.2

Introduction

Stratix® series, Cyclone™ series, APEX™ II, APEX 20K (including APEX 20KE and APEX 20KC), Mercury™, ACEX® 1K, FLEX® 10K (including FLEX 10KE and FLEX 10KA), and FLEX 6000 devices can be configured using one of seven configuration schemes. Table 1–1 shows which device families support which configuration schemes.

Table 1–1. Configuration Scheme Device Family Support

Configuration Scheme	Device Family													
	Stratix IV	Stratix III	Stratix II, Stratix II GX	Stratix, Stratix GX	Arria GX	Cyclone III	Cyclone II	Cyclone	APEX II	APEX 20K, APEX 20KE,	Mercury	ACEX 1K	FLEX 10K, FLEX 10KE,	FLEX 6000
Passive Serial (PS)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Active Serial (AS)	✓	✓	✓	—	✓	✓	✓	✓	—	—	—	—	—	—
Active Parallel (AP)	—	—	—	—	—	✓	—	—	—	—	—	—	—	—
Fast Passive Parallel (FPP)	✓	✓	✓	✓	✓	✓	—	—	✓	—	—	—	—	—
Passive Parallel Synchronous (PPS)	—	—	—	—	—	—	—	—	—	✓	✓	✓	✓	—
Passive Parallel Asynchronous (PPA)	—	—	✓	✓	✓	—	—	—	✓	✓	✓	✓	✓	—
Passive Serial Asynchronous (PSA)	—	—	—	—	—	—	—	—	—	—	—	—	—	✓
Joint Test Action Group (JTAG)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	*(1)*

Note to Table 1–1:

(1) Although you cannot configure FLEX 6000 devices through the JTAG pins, you can perform JTAG boundary-scan testing.

All configuration schemes use either an intelligent host or a configuration device(s) (Table 1–2).

Table 1–2. Configuration Schemes

Configuration Scheme	Typical Use
Passive Serial (PS)	Configuration with the enhanced configuration devices (EPC16, EPC8, and EPC4), EPC2, EPC1, EPC1441 configuration devices, serial synchronous microprocessor interface, the USB Blaster USB Port Download Cable, MasterBlaster™ communications cable, ByteBlaster™ II parallel download cable or ByteBlasterMV™ parallel port download cable.
Active Serial (AS)	Configuration with the serial configuration devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128).
Active Parallel (AP)	Configuration with the parallel flash memory which 16 bits of configuration data are loaded on every clock cycle. Sixteen times faster than AS.
Passive Parallel Synchronous (PPS)	Configuration with a parallel synchronous microprocessor interface.
Fast Passive Parallel (FPP)	Configuration with an enhanced configuration device or parallel synchronous microprocessor interface where 8 bits of configuration data are loaded on every clock cycle. Eight times faster than PPS.
Passive Parallel Asynchronous (PPA)	Configuration with a parallel asynchronous microprocessor interface. In this scheme, the microprocessor treats the target device as memory.
Passive Serial Asynchronous (PSA)	Configuration with a serial asynchronous microprocessor interface.
Joint Test Action Group (JTAG)	Configuration through the IEEE Std. 1149.1 (JTAG) pins. (1)

The following chapters discuss how to configure one or more Stratix series, Arria GX series, Cyclone series, APEX II, APEX 20K (including APEX 20KE and APEX 20KC), Mercury, ACEX 1K, FLEX 10K (including FLEX 10KE and FLEX 10KA), and FLEX 6000 devices. The following chapters should be used in conjunction with the following documents:

- Stratix IV (E and GX) Device Handbook
- Stratix III Device Handbook
- Cyclone III Device Handbook
- Arria GX Device Handbook
- Stratix II GX Device Handbook
- Stratix II Device Handbook
- Stratix Device Handbook
- Stratix GX Device Handbook

- Cyclone II Device Handbook
- Cyclone Device Handbook
- APEX II Programmable Logic Device Family Data Sheet
- APEX 20K Programmable Logic Device Family Data Sheet
- APEX 20KC Programmable Logic Device Data Sheet
- Mercury Programmable Logic Device Family Data Sheet
- ACEX 1K Programmable Logic Device Family Data Sheet
- FLEX 10K Embedded Programmable Logic Family Data Sheet
- FLEX 10KE Embedded Programmable Logic Family Data Sheet
- FLEX 6000 Programmable Logic Device Family Data Sheet

Volume I covers how to configure Altera FPGAs, where each chapter covers a different device family. Each subsection describes how to configure the devices with the following configuration schemes:

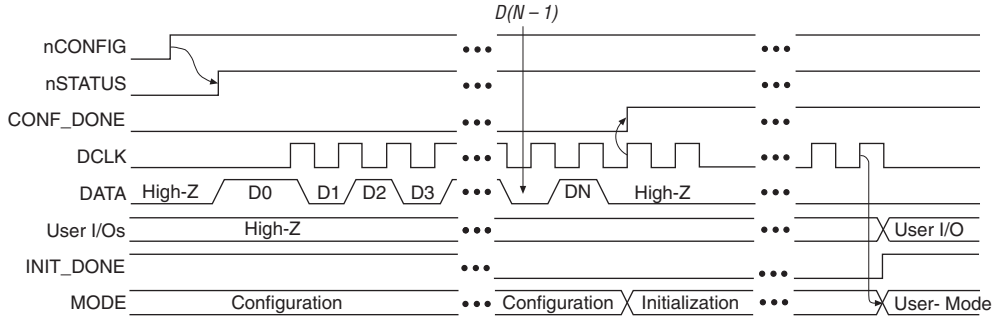
- PS Configuration
  - Using a Configuration Device
  - Using a Microprocessor
  - Using a Download Cable
- AS Configuration
- AP Configuration
- FPP Configuration
  - Using an enhanced Configuration Device
  - Using a Microprocessor
- PPS Configuration
- PPA Configuration
- PSA Configuration
- JTAG Programming and Configuration

Volume II contains information that is relevant for all Altera FPGAs discussed in this handbook. Information about configuration devices and combining different Altera device families in the same configuration chain can be found in this volume.

Device Configuration Overview for Passive Schemes

During device operation, Altera FPGAs store configuration data in SRAM cells. Because SRAM memory is volatile, the SRAM cells must be loaded with configuration data each time the device powers up. After the device is configured, its registers and I/O pins must be initialized. After initialization, the device enters user mode for in-system operation. Figure 1–1 shows the waveform of the configuration pins during configuration, initialization, and user-mode.

Figure 1–1. Configuration Cycle Waveform



The low-to-high transition of nCONFIG on the FPGA begins the configuration cycle. The configuration cycle consists of 3 stages: reset, configuration, and initialization. While nCONFIG is low, the device is in reset. When the device comes out of reset, nCONFIG must be at a logic high level in order for the device to release the open-drain nSTATUS pin. Once nSTATUS is released, it is pulled high by a pull-up resistor and the FPGA is ready to receive configuration data. Before and during configuration all user I/O pins are tri-stated. Stratix series, Cyclone series, APEX II, APEX 20K, Mercury, ACEX 1K, and FLEX 10KE devices have weak pull-up resistors on the I/O pins which are on before and during configuration.

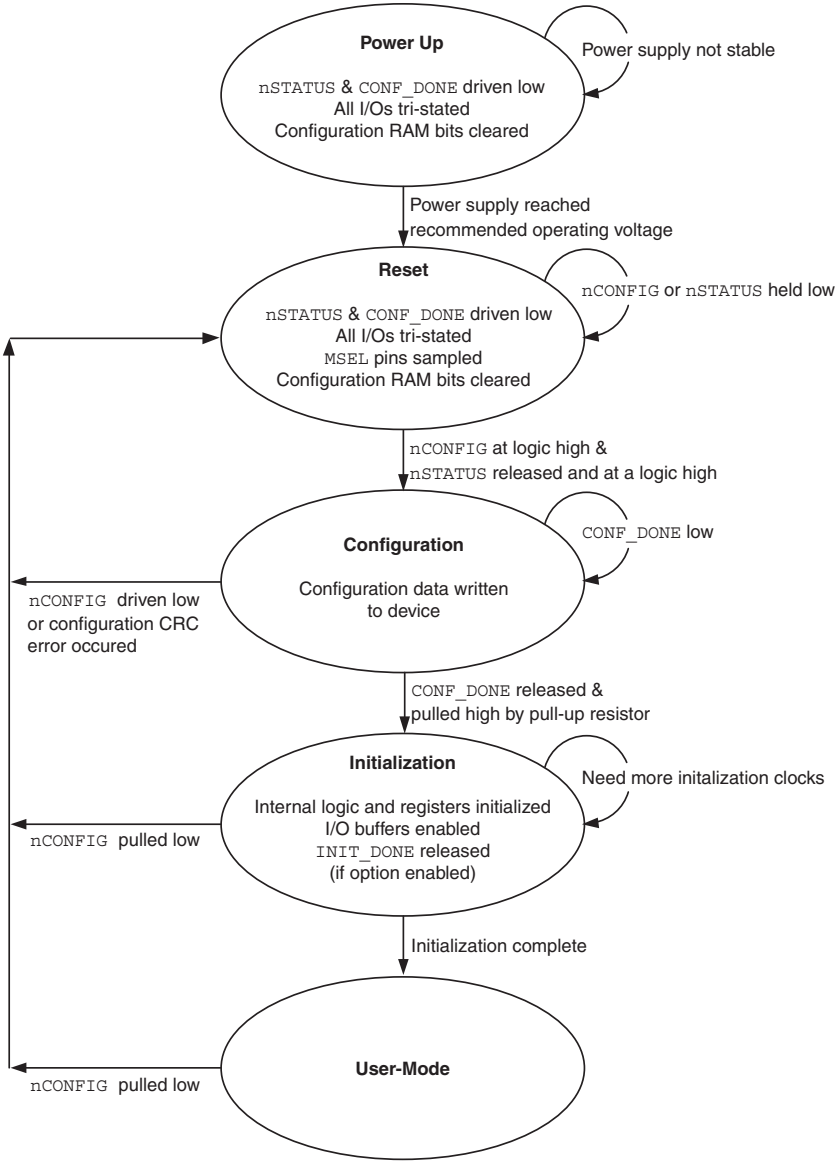
nCONFIG and nSTATUS must be at a logic high level in order for the configuration stage to begin. Configuration can be delayed by holding the nCONFIG low. The device receives configuration data on its DATA pin(s) and (for synchronous configuration schemes) the clock source on the DCLK pin. Configuration data is latched into the FPGA on the rising edge of DCLK. After the FPGA has received all configuration data successfully it releases the CONF_DONE pin, which is pulled high by a pull-up resistor. A low to high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode. During initialization, internal logic, internal and I/O registers are initialized and I/O buffers are enabled. When initialization is finished, the INIT_DONE pin is released and pulled high by an external pull-up resistor. Once in user-mode, the user I/O pins will no longer have a weak pull-up and will function as assigned in your design. The DCLK, DATA (FLEX 6000), and DATA0 (Stratix series, Cyclone series, APEX II, APEX 20K, Mercury, ACEX 1K, and FLEX 10KE) pins should not be left floating after configuration; they should be driven high or low, whichever is convenient, on your board.

A reconfiguration is initiated by toggling the nCONFIG pin from high to low and then back to high. When nCONFIG is pulled low, nSTATUS and CONF_DONE are also pulled low and all I/O pins are tri-stated. Once nCONFIG and nSTATUS return to a logic high level, configuration begins.

Figure 1–2 shows a simple state diagram of the configuration process.

Figure 1–2. Configuration Cycle State Machine



Selecting a Configuration Scheme

The configuration data for Altera devices can be loaded using an active, passive or JTAG configuration scheme. When using an active configuration scheme with a serial configuration device, the target FPGA generates the control and synchronization signals. When both devices are ready to begin configuration, the serial configuration device sends data to the FPGA.

When using any passive configuration scheme, the Altera device is incorporated into a system with an Altera configuration device or an intelligent host, such as a microprocessor, that controls the configuration process. The configuration device or host supplies configuration data from a storage device (a configuration device(s), a hard disk, RAM, or other system memory). When using passive configuration, you can change the target device's functionality while the system is in operation by reconfiguring it.

Altera devices support a number of configuration schemes. Not all device families support all configuration schemes. Table 1–1 and the individual device family sections should be referenced to determine if your target device family supports your intended configuration scheme. Once you have decided on the appropriate configuration scheme for your system, you will need to drive the dedicated mode select control pins, MSEL, of the FPGA to set the configuration mode.

For further details on how to set the MSEL pins for your target device, refer to the appropriate device family chapters.

Below is a brief description of each configuration scheme. For detailed information, consult the appropriate sections.

For supported devices for each configuration scheme, refer Table 1–1 on page 1.

Passive Serial Configuration

PS configuration can be performed by using an Altera download cable, an Altera enhanced configuration device or configuration device, or an intelligent host, such as a microprocessor. During PS configuration, configuration data is transferred from a storage device, such as a configuration device or flash memory, to the FPGA on the DATA (FLEX 6000) or DATA 0 (Stratix series, Cyclone series, APEX II, APEX 20K, Mercury, APCEX 1K, and FLEX 10K) pin. This configuration data is latched into the FPGA on the rising edge of DCLK. Configuration data is transferred one bit per clock cycle.

Active Serial Configuration

AS configuration can be performed by using an Altera Serial Configuration device. During AS configuration, the FPGA device is the master and the configuration device is the slave. Configuration data is transferred to the FPGA on the DATA 0 pin. This configuration data is synchronized to the DCLK input. Configuration data is transferred one bit per clock cycle.

Active Parallel Configuration

In the AP configuration scheme, Altera FPGA devices are configured using commodity 16-bit parallel flash memory. These external non-volatile configuration devices are industry standard microprocessor flash memories. During AP configuration, the FPGA device is the master and the parallel flash memory is the slave. Configuration data is transferred to the Altera device on the DATA[15:0] pins. This configuration data is synchronized to the DCLK input. Configuration data is transferred at a rate of 16 bits per clock cycle. The DCLK frequency driven out by the Altera device during AP configuration is approximately 40 MHz.

Passive Parallel Synchronous Configuration

PPS configuration can be performed by using an intelligent host, such as a microprocessor. During PPS configuration, configuration data is transferred from a storage device, such as flash memory, to the FPGA on the DATA[7..0] pins. This configuration data is synchronized to the DCLK input. On the first rising edge of DCLK, a byte of configuration data is latched into the FPGA. The next 8 falling edges of DCLK are needed to internally serialize the data in the FPGA.

Fast Passive Parallel Configuration

FPP configuration can be performed by using an Altera enhanced configuration device, or an intelligent host, such as a microprocessor. During FPP configuration, configuration data is transferred from a storage device, such as an enhanced configuration device or flash memory, to the FPGA on the DATA[7..0] pins. This configuration data is latched into the FPGA on the rising edge of DCLK. Configuration data is transferred one byte per clock cycle.

Passive Parallel Asynchronous Configuration

PPA configuration can be performed by using an intelligent host, such as a microprocessor. During PPA configuration, configuration data is transferred from a storage device, such as a configuration device or flash memory, to the FPGA on the DATA[7..0] pins. Since this configuration scheme is asynchronous, control signals are used to regulate the configuration cycle.

Passive Serial Asynchronous Configuration

PSA configuration can be performed by using an intelligent host, such as a microprocessor. During PSA configuration, configuration data is transferred from a storage device, such as a configuration device or flash memory, to the FPGA on the DATA pin. Since this configuration scheme is asynchronous, control signals are used to regulate the configuration cycle.

JTAG Configuration

JTAG configuration uses the IEEE Std 1149.1 JTAG interface pins and supports the JAM STAPL standard. JTAG configuration can be performed by using an Altera download cable or an intelligent host, such as a microprocessor.

Document Revision History

Table 1–3 shows the revision history for this document.

Table 1–3. Document Revision History

Date & Document Version	Changes Made	Summary of Changes
October 2008 v2.2	• Updated Table 1–1 and Table 1–2. • Updated Figure 1–2. • Updated "Introduction", "Device Configuration Overview for Passive Schemes", "Selecting a Configuration Scheme", "Passive Serial Configuration", "Active Serial Configuration", "Passive Parallel Synchronous Configuration", "Fast Passive Parallel Configuration", "Passive Parallel Asynchronous Configuration", "Passive Serial Asynchronous Configuration", and "JTAG Configuration" sections. • Added "Active Parallel Configuration" and "Document Revision History" sections.	—

6. Configuring APEX II Devices

CF51004-2.1

Introduction

APEX™ II devices can be configured using one of four configuration schemes. All configuration schemes use either a microprocessor or configuration device.

APEX II devices can be configured using the passive serial (PS), fast passive parallel (FPP), passive parallel asynchronous (PPA), and Joint Test Action Group (JTAG) configuration schemes. The configuration scheme used is selected by driving the APEX II device MSEL1 and MSEL0 pins either high or low as shown in Table 6–1. If your application only requires a single configuration mode, the MSEL pins can be connected to V_{CC} (V_{CCIO} of the I/O bank where the MSEL pin resides) or to ground. If your application requires more than one configuration mode, you can switch the MSEL pins after the FPGA is configured successfully. Toggling these pins during user-mode does not affect the device operation; however, the MSEL pins must be valid before a reconfiguration is initiated.

Table 6–1. APEX II Configuration Schemes

MSEL1	MSEL0	Configuration Scheme
0	0	PS
1	0	FPP
1	1	PPA
(1)	*(1)*	JTAG Based *(2)*

Notes to Table 6–1:

(1) Do not leave the MSEL pins floating; connect them to a low- or high-logic level. These pins support the non-JTAG configuration scheme used in production. If only JTAG configuration is used, you should connect the MSEL pins to ground.

(2) JTAG-based configuration takes precedence over other configuration schemes, which means MSEL pin settings are ignored.

Table 6–2 shows the approximate configuration file sizes for APEX II devices.

Table 6–2. APEX II Raw Binary File (.rbf) Sizes

Device	Data Size (Bits)	Data Size (Bytes)
EP2A15	4,358,512	544,814
EP2A25	6,275,200	784,400
EP2A40	9,640,528	1,208,320
EP2A70	17,417,088	2,177,136

Use the data in Table 6–2 only to estimate the file size before design compilation. Different configuration file formats, such as a Hexidecimal (**.hex**) or Tabular Text File (**.ttf**) format, will have different file sizes. However, for any specific version of the Quartus® II or MAX+PLUS® II software, all designs targeted for the same device will have the same configuration file size.

The following chapter describes in detail how to configure APEX II devices using the supported configuration schemes. The last section describes the device configuration pins available. In this chapter, the generic term device(s) or FPGA(s) will include all APEX II devices.

For more information on setting device configuration options or creating configuration files, refer to *Software Settings*, chapter 6 and 7 n volume 2 of the *Configuration Handbook*.

Passive Serial Configuration

You can perform APEX II PS configuration using an Altera configuration device, an intelligent host (e.g., a microprocessor or Altera® MAX® device), or a download cable.

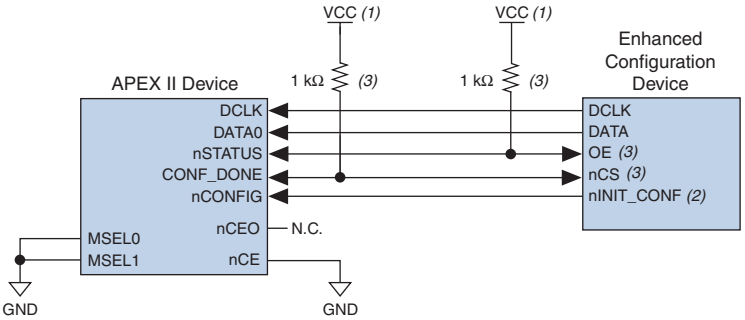
PS Configuration Using a Configuration Device

You can use an Altera configuration device, such as an enhanced configuration device, EPC2, or EPC1 device, to configure APEX II devices using a serial configuration bitstream. Configuration data is stored in the configuration device. Figure 6–1 shows the configuration interface connections between the APEX II device and a configuration device.

The figures in this chapter only show the configuration-related pins and the configuration pin connections between the configuration device and the FPGA.

For more information on the enhanced configuration device and flash interface pins (e.g., PGM[2..0], EXCLK, PORSEL, A[20..0], and DQ[15..0]), refer to the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* in the *Configuration Handbook*.

Figure 6–1. Single Device PS Configuration Using an Enhanced Configuration Device



Notes to Figure 6–1:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) has an internal pull-up resistor that is always active, meaning an external pull-up resistor is not required on the nINIT_CONF/nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the *Disable nCS and OE pull-ups on configuration device* option when generating programming files.

The value of the internal pull-up resistors on the enhanced configuration devices and EPC2 devices can be found in the Operating Conditions table of the Enhanced Configuration Devices (EPC4, EPC8, & EPC16) Data Sheet or the Configuration Devices for SRAM-based LUT Devices Data Sheet.

When using enhanced configuration devices or EPC2 devices, nCONFIG of the FPGA can be connected to nINIT_CONF, which allows the INIT_CONF JTAG instruction to initiate FPGA configuration. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CC} either directly or through a resistor. An

internal pull-up on the nINIT_CONF pin is always active in enhanced configuration devices and EPC2 devices, which means an external pull-up resistor is not required if nCONFIG is tied to nINIT_CONF.

Upon power-up, the APEX II device goes through a Power-On Reset (POR) for approximately 5 µs. During POR, the device resets and holds nSTATUS low, and tri-states all user I/O pins. The configuration device also goes through a POR delay to allow the power supply to stabilize. The POR time for EPC2, EPC1, and EPC1441 devices is 200 ms (maximum), and for enhanced configuration devices, the POR time can be set to either 100 ms or 2 ms, depending on its PORSEL pin setting. If the PORSEL pin is connected to GND, the POR delay is 100 ms. During this time, the configuration device drives its OE pin low. This low signal delays configuration because the OE pin is connected to the target device's nSTATUS pin. When both devices complete POR, they release their open-drain OE or nSTATUS pin, which is then pulled high by a pull-up resistor. Once the FPGA successfully exits POR, all user I/O pins are tri-stated. APEX II devices have weak pull-up resistors on the user I/O pins which are on before and during configuration.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the Operating Conditions table of the *APEX II Programmable Logic Device Family Data Sheet*.

When the power supplies have reached the appropriate operating voltages, the target FPGA senses the low-to-high transition on nCONFIG and initiates the configuration cycle. The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. The beginning of configuration can be delayed by holding the nCONFIG or nSTATUS pin low.

VCCINT and VCCIO pins on the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the nSTATUS pin, which is pulled high by a pull-up resistor. Enhanced configuration and EPC2 devices have an optional internal pull-up on the OE pin. This option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If this internal pull-up resistor is not used, an external 1-kΩ pull-up resistor on the OE/nSTATUS line is required. Once nSTATUS is released, the FPGA is ready to receive configuration data and the configuration stage begins.

When nSTATUS is pulled high, OE of the configuration device also goes high and the configuration device clocks data out serially to the FPGA using its internal oscillator. The APEX II device receives configuration data on its DATA0 pin and the clock is received on the DCLK pin. Data is latched into the FPGA on the rising edge of DCLK.

After the FPGA has received all configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by a pull-up resistor. Since CONF_DONE is tied to the configuration device's nCS pin, the configuration device is disabled when CONF_DONE goes high. Enhanced configuration and EPC2 devices have an optional internal pull-up resistor on the nCS pin. This option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If this internal pull-up is not used, an external 1-kΩpull-up resistor on the nCS/CONF_DONE line is required. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin.

In APEX II devices, the initialization clock source is either the APEX II internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the APEX II device will supply itself with enough clock cycles for proper initialization. You also have the flexibility to synchronize initialization of multiple devices by using the CLKUSR option. You can turn on the *Enable user-supplied start-up clock (CLKUSR)* option in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR will not affect the configuration process. After all configuration data is accepted and CONF_DONE goes high, APEX II devices require 40 clock cycles to properly initialize.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The *Enable INIT_DONE output* option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used, it will be high due to an external 1-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. This low-to-high transition signals that the FPGA has entered user mode. In user-mode, the user I/O pins will no longer have weak pull-up resistors and will function as assigned in your design. The enhanced configuration device drives DCLK low and DATA high at the end of configuration.

If an error occurs during configuration, the FPGA drives its nSTATUS pin low, resetting itself internally. Since the nSTATUS pin is tied to OE, the configuration device will also be reset. If the *Auto-Restart Configuration After Error* option available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box is turned on, the FPGA automatically initiates reconfiguration if an error occurs. The APEX II device will release its nSTATUS pin after a reset time-out period (maximum of 40 µs). When the nSTATUS pin is released and pulled high by a pull-up resistor, the configuration device reconfigures the chain. If this option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 8 µs to restart configuration. The external system can pulse nCONFIG if nCONFIG is under system control rather than tied to V_{CC}.

In addition, if the configuration device sends all of its data and then detects that CONF_DONE has not gone high, it recognizes that the FPGA has not configured successfully. Enhanced configuration devices wait for 64 DCLK cycles after the last configuration bit was sent for CONF_DONE to reach a high state. EPC2 devices wait for 16 DCLK cycles. In this case, the configuration device pulls its OE pin low, which in turn drives the target device's nSTATUS pin low. If the *Auto-Restart Configuration After Error* option is set in the software, the target device resets and then releases its nSTATUS pin after a reset time-out period (maximum of 40 µs). When nSTATUS returns high, the configuration device tries to reconfigure the FPGA.

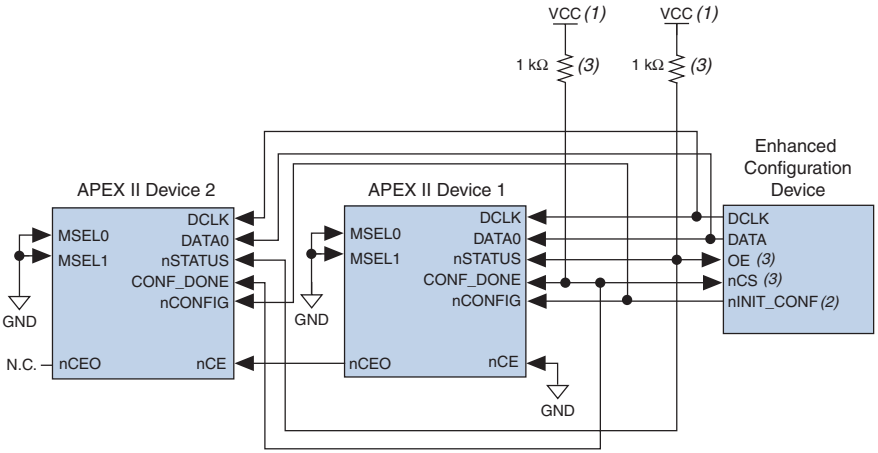
When CONF_DONE is sensed low after configuration, the configuration device recognizes that the target device has not configured successfully; therefore, your system should not pull CONF_DONE low to delay initialization. Instead, use the *CLKUSR* option to synchronize the initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain will initialize together if their CONF_DONE pins are tied together.

If the optional CLKUSR pin is being used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure that CLKUSR continues toggling during the time nSTATUS is low (maximum of 40 µs).

When the FPGA is in user-mode, a reconfiguration can be initiated by pulling the nCONFIG pin low. The nCONFIG pin should be low for at least 8 µs. When nCONFIG is pulled low, the FPGA also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Since CONF_DONE is pulled low, this will activate the configuration device since it will see its nCS pin drive low. Once nCONFIG returns to a logic high state and nSTATUS is released by the FPGA, reconfiguration begins.

Figure 6–2 shows how to configure multiple devices with a configuration device. This circuit is similar to the configuration device circuit for a single device, except APEX II devices are cascaded for multi-device configuration.

Figure 6–2. Multi-Device PS Configuration Using an Enhanced Configuration Device



Notes to Figure 6–2:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) has an internal pull-up resistor that is always active, meaning an external pull-up resistor is not required on the nINIT_CONF/nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the *Disable nCS and OE pull-ups on configuration device* option when generating programming files.

Enhanced configuration devices (EPC4, EPC8, and EPC16) cannot be cascaded.

When performing multi-device configuration, you must generate the configuration device's Programmer Object File (**.pof**) from each project's SRAM Object File (**.sof**). You can combine multiple SOFs using the Quartus II software.

For more information on how to create configuration files for multi-device configuration chains, *refer to Software Settings*, chapter 6 and 7 in volume 2 of the *Configuration Handbook*.

In multi-device PS configuration, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they can require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DCLK and DATA lines are buffered for every fourth device.

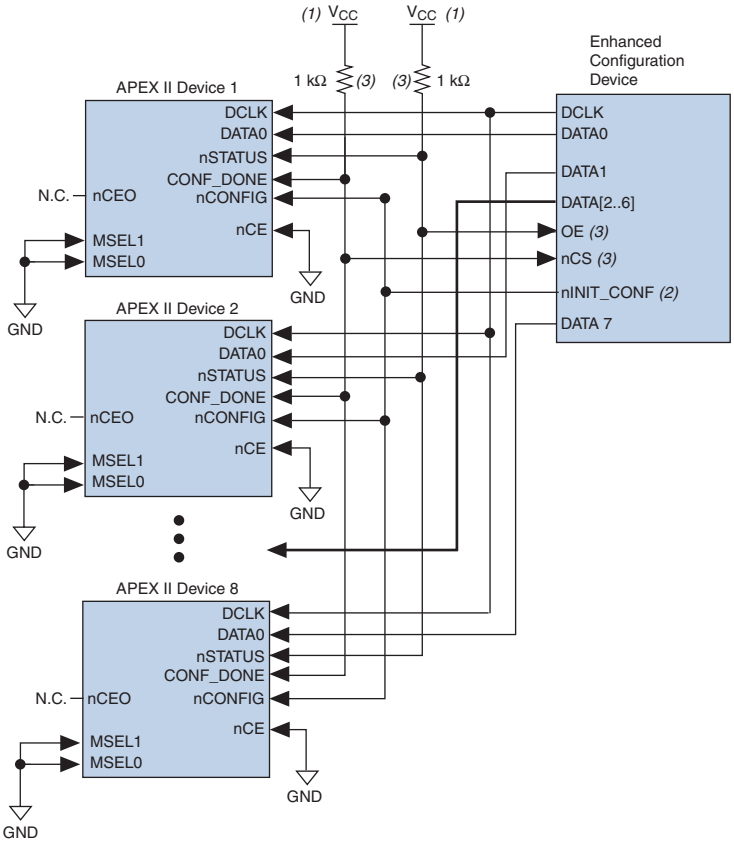
When configuring multiple devices, configuration does not begin until all devices release their OE or nSTATUS pins. Similarly, since all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Since all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first FPGA flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This low signal drives the OE pin low on the enhanced configuration device and drives nSTATUS low on all FPGAs, which causes them to enter a reset state. This behavior is similar to a single FPGA detecting an error.

If the *Auto-Restart Configuration After Error* option is turned on, the devices will automatically initiate reconfiguration if an error occurs. The FPGAs will release their nSTATUS pins after a reset time-out period (maximum of 40 µs). When all the nSTATUS pins are released and pulled high, the configuration device tries to reconfigure the chain. If the *Auto-Restart Configuration After Error* option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 8 µs to restart configuration. The external system can pulse nCONFIG if nCONFIG is under system control rather than tied to V_{CC}.

The enhanced configuration devices also support parallel configuration of up to eight devices. The n-bit (n = 1, 2, 4, or 8) PS configuration mode allows enhanced configuration devices to concurrently configure FPGAs or a chain of FPGAs. In addition, these devices do not have to be the same device family or density; they can be any combination of Altera FPGAs. An individual enhanced configuration device DATA line is available for each targeted FPGA. Each DATA line can also feed a daisy chain of FPGAs. Figure 6–3 shows how to concurrently configure multiple devices using an enhanced configuration device.

Figure 6–3. Concurrent PS Configuration of Multiple Devices Using an Enhanced Configuration Device



Notes to Figure 6–3:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) has an internal pull-up resistor that is always active, meaning an external pull-up resistor is not required on the nINIT_CONF/nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the *Disable nCS and OE pull-ups on configuration device* option when generating programming files.

The Quartus II software only allows the selection of n-bit PS configuration modes, where n must be 1, 2, 4, or 8. However, you can use these modes to configure any number of devices from 1 to 8. When configuring SRAM-based devices using n-bit PS modes, use Table 6–3 to select the appropriate configuration mode for the fastest configuration times.

Table 6–3. Recommended Configuration Using n-Bit PS Modes

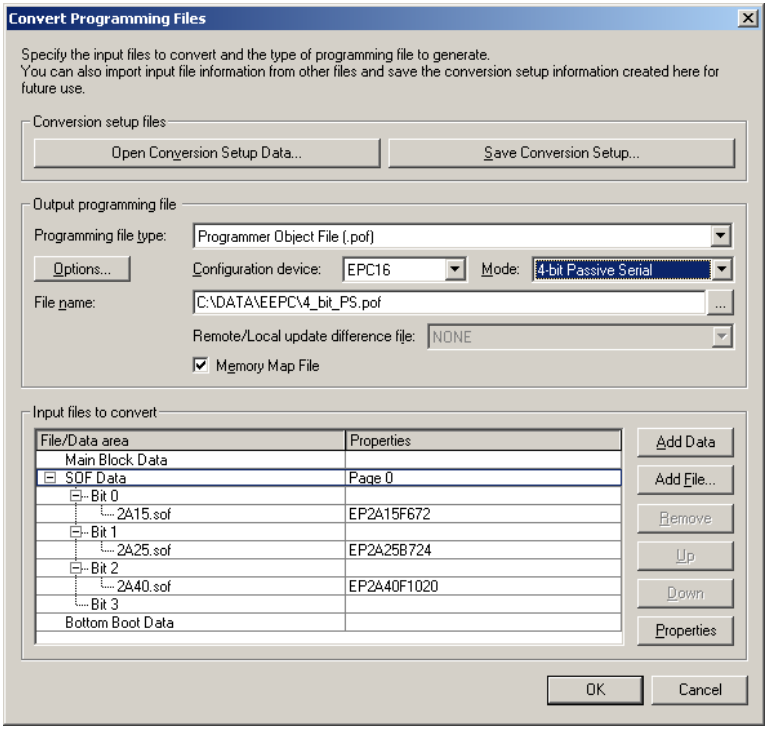
Number of Devices *(1)*	Recommended Configuration Mode
1	1-bit PS
2	2-bit PS
3	4-bit PS
4	4-bit PS
5	8-bit PS
6	8-bit PS
7	8-bit PS
8	8-bit PS

Note to Table 6–3:

(1) Assume that each DATA line is only configuring one device, not a daisy chain of devices.

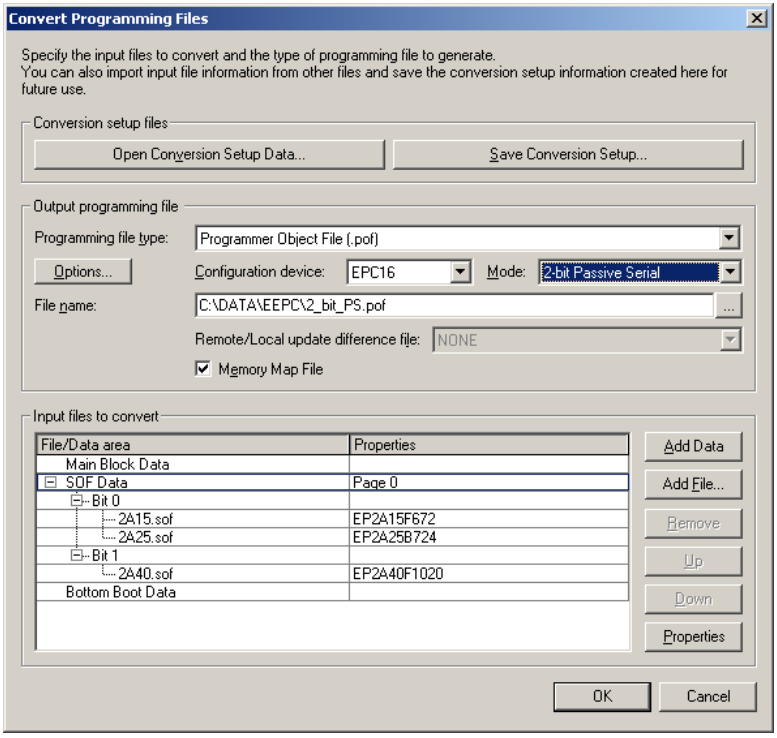
For example, if you configure three FPGAs, you would use the 4-bit PS mode. For the DATA0, DATA1, and DATA2 lines, the corresponding SOF data is transmitted from the configuration device to the FPGA. For DATA3, you can leave the corresponding Bit3 line blank in the Quartus II software. On the printed circuit board (PCB), leave the DATA3 line from the enhanced configuration device unconnected. Figure 6–4 shows the Quartus II **Convert Programming Files** window (Tools menu) setup for this scheme.

Figure 6–4. Software Settings for Configuring Devices Using n-Bit PS Modes



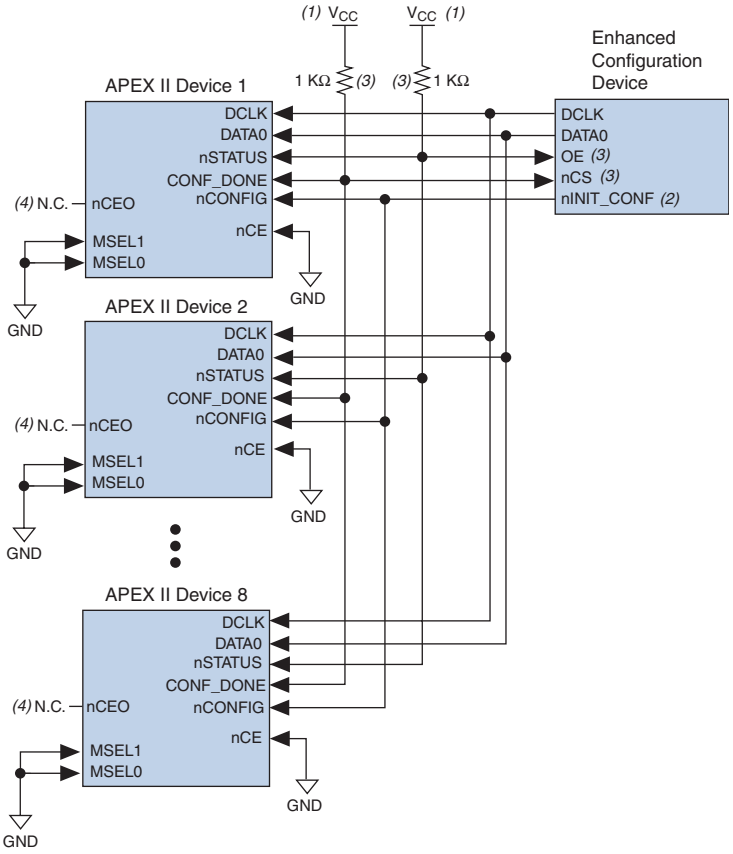
Alternatively, you can daisy chain two FPGAs to one DATA line while the other DATA lines drive one device each. For example, you could use the 2-bit PS mode to drive two FPGAs with DATA Bit0 (EP2A15 and EP2A25 devices) and the third device (the EP2A40 device) with DATA Bit1. This 2-bit PS configuration scheme requires less space in the configuration flash memory, but can increase the total system configuration time. See Figure 6–5.

Figure 6–5. Software Settings for Daisy Chaining Two FPGAs on One DATA Line



In your system, you may have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they can require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete configuration at the same time. Figure 6–6 shows multi-device PS configuration when the APEX II devices are receiving the same configuration data.

Figure 6–6. Multiple-Device PS Configuration Using an Enhanced Configuration Device When FPGAs Receive the Same Data



Notes to Figure 6–6:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) has an internal pull-up resistor that is always active, meaning an external pull-up resistor is not required on the nINIT_CONF/nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the *Disable nCS and OE pull-ups on configuration device* option when generating programming files.

(4) The nCEO pins of all devices are left unconnected when configuring the same configuration data into multiple devices.

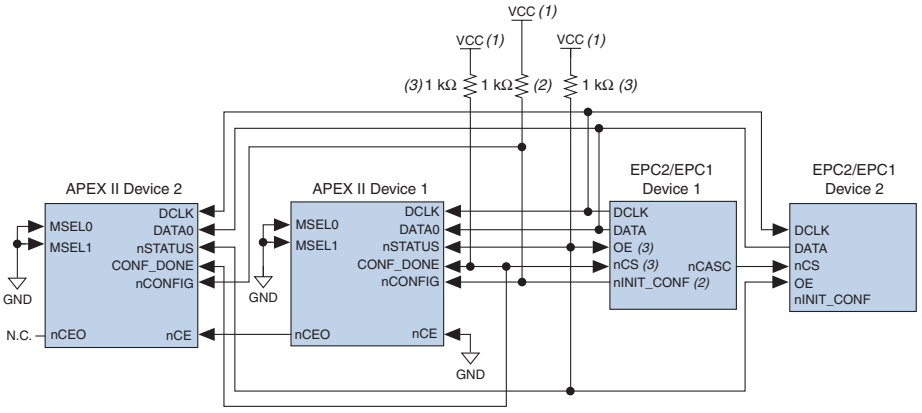
You can cascade several EPC2 or EPC1 devices to configure multiple APEX II devices. The first configuration device in the chain is the master configuration device, while the subsequent devices are the slave devices. The master configuration device sends DCLK to the APEX II devices and to the slave configuration devices. The first EPC device's nCS pin is connected to the CONF_DONE pins of the FPGAs, while its nCASC pin is connected to nCS of the next configuration device in the chain. The last device's nCS input comes from the previous device, while its nCASC pin is left floating. When all data from the first configuration device is sent, it drives nCASC low, which in turn drives nCS on the next configuration device. Because a configuration device requires less than one clock cycle to activate a subsequent configuration device, the data stream is uninterrupted.

Enhanced configuration devices EPC4, EPC8, and EPC16 cannot be cascaded.

Since all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, the master configuration device stops configuration for the entire chain and the entire chain must be reconfigured. For example, if the master configuration device does not detect CONF_DONE going high at the end of configuration, it resets the entire chain by pulling its OE pin low. This low signal drives the OE pin low on the slave configuration device(s) and drives nSTATUS low on all FPGAs, causing them to enter a reset state. This behavior is similar to the FPGA detecting an error in the configuration data.

Figure 6–7 shows how to configure multiple devices using cascaded EPC2 or EPC1 devices.

Figure 6–7. Multi-Device PS Configuration Using Cascaded EPC2 or EPC1 Devices



Notes to Figure 6–7:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) has an internal pull-up resistor that is always active, meaning an external pull-up resistor is not required on the nINIT_CONF/nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the *Disable nCS and OE pull-ups on configuration device* option when generating programming files.

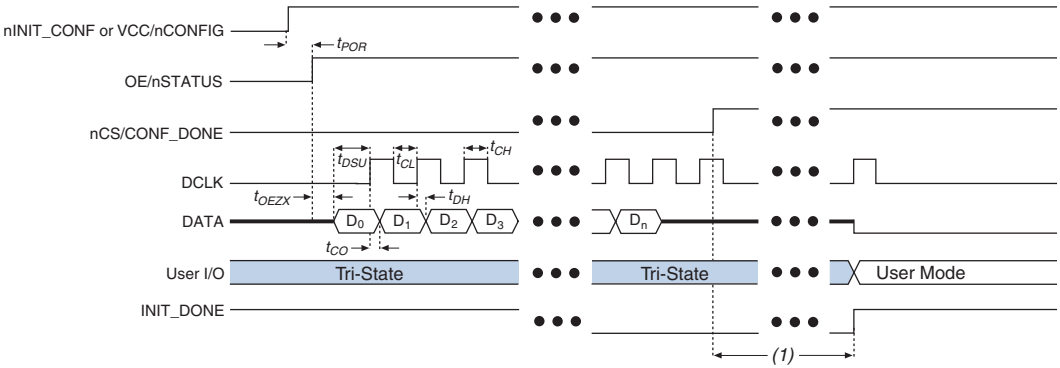
When using enhanced configuration devices or EPC2 devices, nCONFIG of the FPGA can be connected to nINIT_CONF, which allows the INIT_CONF JTAG instruction to initiate FPGA configuration. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CC} either directly or through a resistor. An internal pull-up resistor on the nINIT_CONF pin is always active in the enhanced configuration devices and the EPC2 devices, which means an external pull-up is not required if nCONFIG is tied to nINIT_CONF. If multiple EPC2 devices are used to configure an APEX II device(s), only the first EPC2 has its nINIT_CONF pin tied to the device's nCONFIG pin.

You can use a single configuration chain to configure APEX II devices with other Altera devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information on configuring multiple Altera devices in the same configuration chain, refer to *Configuring Mixed Altera FPGA Chains* chapter 8 in volume 2 of the *Configuration Handbook*.

Figure 6–8 shows the timing waveform for the PS configuration scheme using a configuration device.

Figure 6–8. APEX II PS Configuration Using a Configuration Device Timing Waveform



Note to Figure 6–8:

(1) APEX II devices enter user-mode 40 clock cycles after `CONF_DONE` goes high. The initialization clock can come from the APEX II internal oscillator or the `CLKUSR` pin.

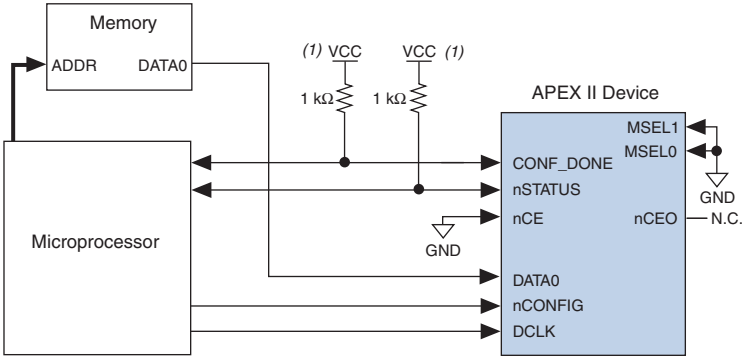
For timing information, refer to the *Enhanced Configuration Devices (EPC4, EPC8, and EPC16) Data Sheet* or the *Configuration Devices for SRAM-based LUT Devices Data Sheet* in the *Configuration Handbook*.

Device configuration options and how to create configuration files are discussed further in *Section II, Software Settings*, chapter 6 and 7 in volume 2 of the *Configuration Handbook*.

PS Configuration Using a Microprocessor

In the PS configuration scheme, an intelligent host (e.g., a microprocessor or CPLD) can transfer configuration data from a storage device (e.g., flash memory) to the target APEX II devices. Configuration data can be stored in RBF, HEX, or TTF format. Figure 6–9 shows the configuration interface connections between the APEX II device and a microprocessor for single device configuration.

Figure 6–9. Single Device PS Configuration Using a Microprocessor



Note to Figure 6–9:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for the device.

Upon power-up, the APEX II device goes through a POR for approximately 5 µs. During POR, the device resets and holds nSTATUS low, and tri-states all user I/O pins. Once the FPGA successfully exits POR, all user I/O pins are tri-stated. APEX II devices have weak pull-up resistors on the user I/O pins which are on before and during configuration.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the Operating Conditions table of the *APEX II Programmable Logic Device Family Data Sheet*.

The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration, the microprocessor must generate a low-to-high transition on the nCONFIG pin.

VCCINT and VCCIO pins on the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 1-kΩ pull-up resistor. Once nSTATUS is released, the FPGA is ready to receive configuration data and the configuration stage begins. When nSTATUS is pulled high, the microprocessor should place the configuration data one bit at a time on the DATA0 pin. The least significant bit (LSB) of each data byte must be sent first.

The APEX II device receives configuration data on its DATA0 pin and the clock is received on the DCLK pin. Data is latched into the FPGA on the rising edge of DCLK. Data is continuously clocked into the target device until CONF_DONE goes high. After the FPGA has received all configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by an external 1-kΩ pull-up resistor. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin.

In APEX II devices, the initialization clock source is either the APEX II internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the APEX II device will take care to provide itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device. Driving DCLK to the device after configuration is complete does not affect device operation.

You also have the flexibility to synchronize initialization of multiple devices by using the CLKUSR option. The *Enable user-supplied start-up clock (CLKUSR)* option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR will not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, APEX II devices require 40 clock cycles to initialize properly.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The *Enable INIT_DONE output* option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used it will be high due to an external 1-kΩ pull-up when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low. When initialization is complete, the INIT_DONE pin will be released and pulled high. The microprocessor must be able to detect this low-to-high transition which signals the FPGA has entered user mode. In user-mode, the user I/O pins will no longer have weak pull-up resistors and will function as assigned in your design. To ensure DCLK and DATA are not left floating at the end of configuration, the microprocessor must drive them either high or low, whichever is convenient on your board.

Handshaking signals are not used in PS configuration mode. Therefore, the configuration clock (DCLK) speed must be below the specified frequency to ensure correct configuration. No maximum DCLK period exists, which means you can pause configuration by halting DCLK for an indefinite amount of time.

If an error occurs during configuration, the FPGA drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the microprocessor that there is an error. If the *Auto-Restart Configuration After Error* option (available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box) is turned on, the APEX II device releases nSTATUS after a reset time-out period (maximum of 40 µs). After nSTATUS is released and pulled high by a pull-up resistor, the microprocessor can try to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition (with a low pulse of at least 8 µs) on nCONFIG to restart the configuration process.

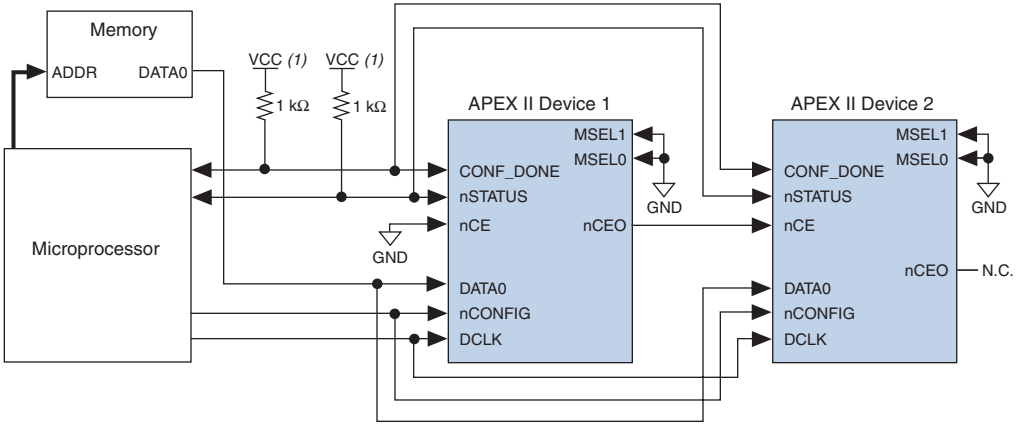
The microprocessor can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The CONF_DONE pin must be monitored by the microprocessor to detect errors and determine when programming completes. If the microprocessor sends all configuration data but CONF_DONE or INIT_DONE have not gone high, the microprocessor must reconfigure the target device.

If the optional CLKUSR pin is being used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure that CLKUSR continues toggling during the time nSTATUS is low (maximum of 40 µs).

When the FPGA is in user-mode, you can initiate a reconfiguration by transitioning the nCONFIG pin low-to-high. The nCONFIG pin must be low for at least 8 µs. When nCONFIG is pulled low, the FPGA also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Once nCONFIG returns to a logic high state and nSTATUS is released by the FPGA, reconfiguration begins.

Figure 6–10 shows how to configure multiple devices using a microprocessor. This circuit is similar to the PS configuration circuit for a single device, except APEX II devices are cascaded for multi-device configuration.

Figure 6–10. Multi-Device PS Configuration Using a Microprocessor



Note to Figure 6–10:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.

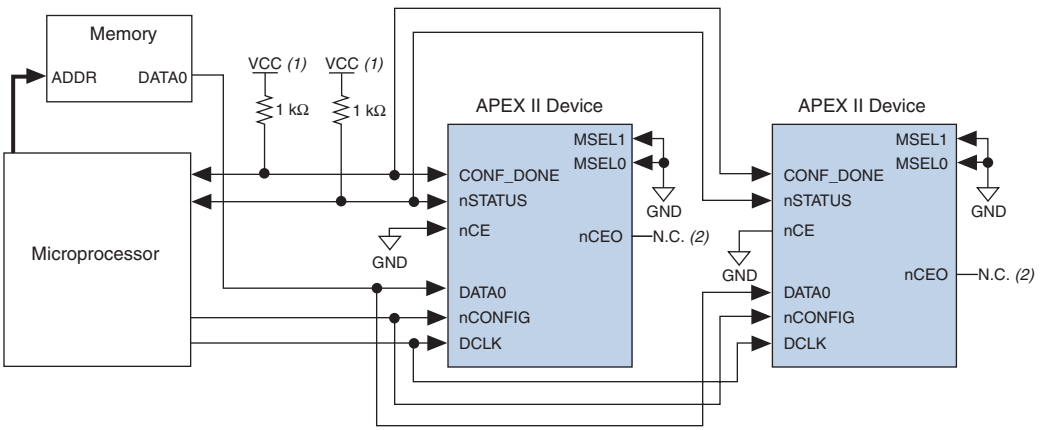
In multi-device PS configuration the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration within one clock cycle; therefore, the transfer of data destinations is transparent to the microprocessor. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they can require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DCLK and DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Since all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first FPGA flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single FPGA detecting an error.

If the *Auto-Restart Configuration After Error* option is turned on, the FPGAs release their nSTATUS pins after a reset time-out period (maximum of 40 µs). After all nSTATUS pins are released and pulled high, the microprocessor can try to reconfigure the chain without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition (with a low pulse of at least 8 µs) on nCONFIG to restart the configuration process.

In your system, you can have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they can require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete configuration at the same time. Figure 6–11 shows multi-device PS configuration when both APEX II devices are receiving the same configuration data.

Figure 6–11. Multiple-Device PS Configuration Using a Microprocessor When Both FPGAs Receive the Same Data



Notes to Figure 6–11:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.

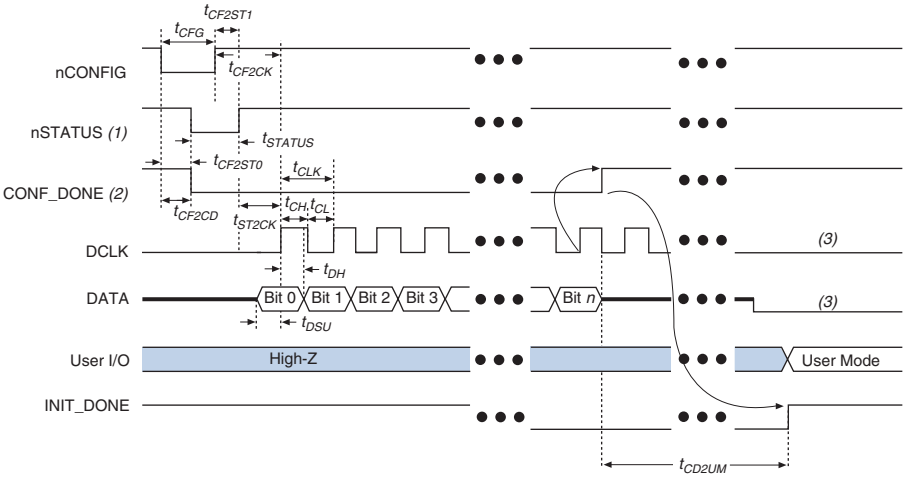
(2) The nCEO pins of both devices are left unconnected when configuring the same configuration data into multiple devices.

You can use a single configuration chain to configure APEX II devices with other Altera devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information on configuring multiple Altera devices in the same configuration chain, refer to *Configuring Mixed Altera FPGA Chains* chapter 8 in volume 2 of the *Configuration Handbook*.

Figure 6–12 shows the timing waveform for the PS configuration for APEX II devices using a microprocessor.

Figure 6–12. APEX II PS Configuration Using a Microprocessor Timing Waveform



Notes to Figure 6–12:

(1) Upon power-up, the APEX II device holds nSTATUS low for not more than 5 µs after V_{CC} reaches its minimum requirement.

(2) Upon power-up, before and during configuration, CONF_DONE is low.

(3) DATA0 and DCLK should not be left floating after configuration. It should be driven high or low, whichever is more convenient.

Table 6–4 defines the timing parameters for APEX II devices for PS configuration.

Table 6–4. PS Timing Parameters for APEX II Devices Note (1)

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		200	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		200	ns
t_{CFG}	nCONFIG low pulse width	8		µs
t_{STATUS}	nSTATUS low pulse width	10	40	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high		1 *(2)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	40		µs
t_{ST2CK}	nSTATUS high to first rising edge on DCLK	1		µs
t_{DSU}	Data setup time before rising edge on DCLK	10		ns
t_{DH}	Data hold time after rising edge on DCLK	0		ns
t_{CH}	DCLK high time	7.5		ns
t_{CL}	DCLK low time	7.5		ns
t_{CLK}	DCLK period	15		ns
f_{MAX}	DCLK maximum frequency		66	MHz
t_{CD2UM}	CONF_DONE high to user mode *(3)*	2	8	µs

Notes to Table 6–4:
(1) This information is preliminary.
(2) This value is applicable if users do not delay configuration by extending the nSTATUS low pulse width.
(3) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting the device. If the clock source is CLKUSR, multiply the clock period by 40 for APEX II devices to obtain this value.

Device configuration options and how to create configuration files are discussed further in the *Software Settings*, chapters 6 and 7 in volume 2 of the *Configuration Handbook*.

Configuring Using the MicroBlaster Driver

The MicroBlaster™ software driver allows you to configure Altera's FPGAs through the ByteBlasterMV cable in PS mode. The MicroBlaster software driver supports a RBF programming input file and is targeted for embedded passive serial configuration. The source code is developed for the Windows NT operating system, although you can customize it to run on other operating systems. For more information on the MicroBlaster software driver, go to the Altera web site *http://www.altera.com).*

PS Configuration Using a Download Cable

In this section, the generic term "download cable" includes the Altera USB Blaster universal serial bus (USB) port download cable, MasterBlaster™ serial/USB communications cable, ByteBlaster™ II parallel port download cable, and the ByteBlasterMV™ parallel port download cable.

In PS configuration with a download cable, an intelligent host (e.g., a PC) transfers data from a storage device to the FPGA via the USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV cable.

Upon power-up, the APEX II device goes through a POR for approximately 5 µs. During POR, the device resets and holds nSTATUS low, and tri-states all user I/O pins. Once the FPGA successfully exits POR, all user I/O pins are tri-stated. APEX II devices have weak pull-up resistors on the user I/O pins which are on before and during configuration.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the Operating Conditions table of the *APEX II Programmable Logic Device Family Data Sheet*.

The configuration cycle consists of 3 stages: reset, configuration and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration in this scheme, the download cable generates a low-to-high transition on the nCONFIG pin.

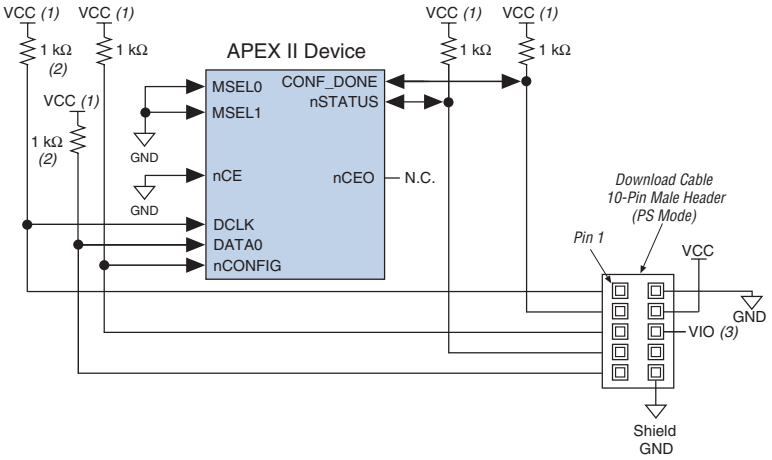
VCCINT and VCCIO pins on the banks where the configuration, and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 1-kΩ pull-up resistor. Once nSTATUS is released the FPGA is ready to receive configuration data and the configuration stage begins. The programming hardware or download cable then places the configuration data one bit at a time on the device's DATA0 pin. The configuration data is clocked into the target device until CONF_DONE goes high.

When using a download cable, setting the *Auto-Restart Configuration After Error* option does not affect the configuration cycle because you must manually restart configuration in the Quartus II software when an error occurs. Additionally, the *Enable user-supplied start-up clock (CLKUSR)* option has no affect on the device initialization since this option is disabled in the SOF when programming the FPGA using the Quartus II programmer and download cable. Therefore, if you turn on the CLKUSR

option, you do not need to provide a clock on CLKUSR when you are configuring the FPGA with the Quartus II programmer and a download cable. Figure 6–13 shows PS configuration for APEX II devices using a USB Blaster, MasterBlaster, ByteBlaster II or ByteBlasterMV cable.

Figure 6–13. PS Configuration Using a USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV Cable



Notes to Figure 6–13:

(1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster (VIO pin), ByteBlaster II or ByteBlasterMV cable.

(2) The pull-up resistors on DATA0 and DCLK are only needed if the download cable is the only configuration scheme used on your board. This is to ensure that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, the pull-up resistors on DATA0 and DCLK are not needed.

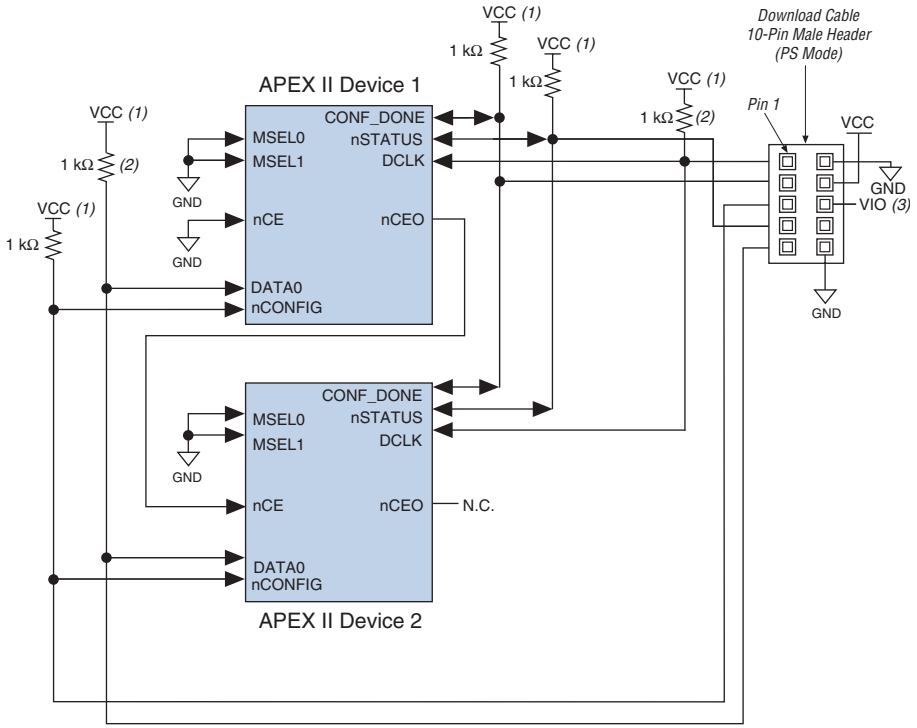
(3) Pin 6 of the header is a VIO reference voltage for the MasterBlaster output driver. VIO should match the device's V_{CCIO}. Refer to the MasterBlaster Serial/USB Communications Cable Data Sheet for this value. In the ByteBlasterMV, this pin is a no connect. In the USB Blaster and ByteBlaster II, this pin is connected to nCE when it is used for Active Serial programming, otherwise it is a no connect.

You can use a download cable to configure multiple APEX II devices by connecting each device's nCEO pin to the subsequent device's nCE pin. The first device's nCE pin is connected to GND while its nCEO pin is connected to the nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. All other configuration pins, nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE are connected to every device in the chain. Because all CONF_DONE pins are tied together, all devices in the chain initialize and enter user mode at the same time.

In addition, because the nSTATUS pins are tied together, the entire chain halts configuration if any device detects an error. The *Auto-Restart Configuration After Error* option does not affect the configuration cycle because you must manually restart configuration in the Quartus II software when an error occurs.

Figure 6–14 shows how to configure multiple APEX II devices with a download cable.

Figure 6–14. Multi-Device PS Configuration using a USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV Cable



Notes to Figure 6–14:

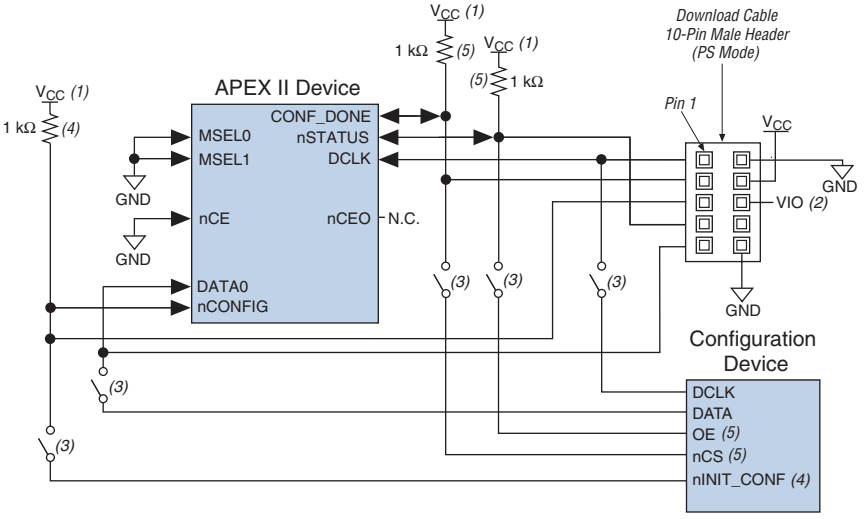
(1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster (VIO pin), ByteBlaster II or ByteBlasterMV cable.

(2) The pull-up resistors on DATA0 and DCLK are only needed if the download cable is the only configuration scheme used on your board. This is to ensure that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, the pull-up resistors on DATA0 and DCLK are not needed.

(3) Pin 6 of the header is a VIO reference voltage for the MasterBlaster output driver. VIO should match the device's V_{CCIO}. Refer to the MasterBlaster Serial/USB Communications Cable Data Sheet for this value. In the ByteBlasterMV, this pin is a no connect. In the USB Blaster and ByteBlaster II, this pin is connected to nCE when it is used for Active Serial programming, otherwise it is a no connect.

If you are using a download cable to configure device(s) on a board that also has configuration devices, you should electrically isolate the configuration device from the target device(s) and cable. One way to isolate the configuration device is to add logic, such as a multiplexer, that can select between the configuration device and the cable. The multiplexer chip should allow bidirectional transfers on the nSTATUS and CONF_DONE signals. Another option is to add switches to the five common signals (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) between the cable and the configuration device. The last option is to remove the configuration device from the board when configuring the FPGA with the cable. Figure 6–15 shows a combination of a configuration device and a download cable to configure an FPGA.

Figure 6–15. PS Configuration with a Download Cable and Configuration Device Circuit



Notes to Figure 6–15:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) Pin 6 of the header is a VIO reference voltage for the MasterBlaster output driver. VIO should match the device's V_{CCIO}. Refer to the MasterBlaster Serial/USB Communications Cable Data Sheet for this value. In the ByteBlasterMV, this pin is a no connect. In the USB Blaster and ByteBlaster II, this pin is connected to nCE when it is used for Active Serial programming, otherwise it is a no connect.

(3) You should not attempt configuration with a download cable while a configuration device is connected to an APEX II device. Instead, you should either remove the configuration device from its socket when using the download cable or place a switch on the five common signals between the download cable and the configuration device.

(4) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) has an internal pull-up resistor that is always active. This means an external pull-up resistor is not required on the nINIT_CONF/nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(5) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the *Disable nCS and OE pull-up resistors on configuration device* option when generating programming files.

For more information on how to use the USB Blaster, MasterBlaster, ByteBlaster II or ByteBlasterMV cables, refer to the following data sheets.

- USB Blaster USB Port Download Cable Data Sheet
- MasterBlaster Serial/USB Communications Cable Data Sheet
- ByteBlaster II Parallel Port Download Cable Data Sheet
- ByteBlasterMV Parallel Port Download Cable Data Sheet

Fast Passive Parallel Configuration

Fast Passive Parallel (FPP) configuration in APEX II devices is designed to meet the continuously increasing demand for faster configuration times. APEX II devices are designed with the capability of receiving byte-wide configuration data per clock cycle, and guarantee a configuration time of less than 100 ms with a 66-MHz configuration clock.

FPP configuration of APEX II devices can be performed using an Altera enhanced configuration device or an intelligent host, such as a microprocessor.

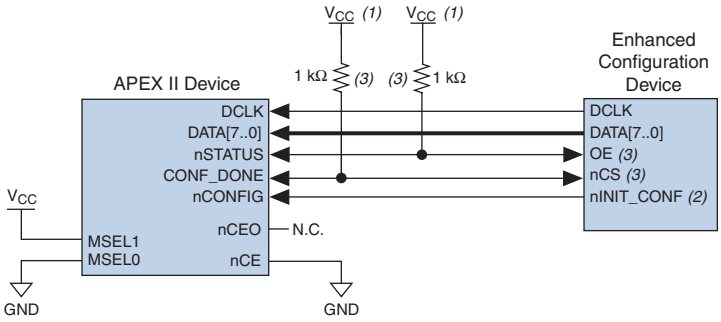
FPP Configuration Using an Enhanced Configuration Device

In the FPP configuration scheme, an enhanced configuration device sends a byte of configuration data every DCLK cycle to the APEX II device. Configuration data is stored in the configuration device. Figure 6–16 shows the configuration interface connections between the APEX II device and the enhanced configuration device for single device configuration.

The figures in this chapter only show the configuration-related pins and the configuration pin connections between the configuration device and the FPGA.

For more information on the enhanced configuration device and flash interface pins, such as PGM[2..0], EXCLK, PORSEL, A[20..0], and DQ[15..0], refer to the *Enhanced Configuration Devices (EPC4, EPC8, & EPC16) Data Sheet* in the *Configuration Handbook.*

Figure 6–16. Single Device FPP Configuration Using an Enhanced Configuration Device



Notes to Figure 6–16:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active. This means an external pull-up resistor is not required on the nINIT_CONF/nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the *Disable nCS and OE pull-ups on configuration device* option when generating programming files.

The value of the internal pull-up resistors on the enhanced configuration devices can be found in the Operating Conditions table of the *Enhanced Configuration Devices (EPC4, EPC8, & EPC16) Data Sheet* in the *Configuration Handbook*.

When using enhanced configuration devices, nCONFIG of the FPGA can be connected to nINIT_CONF, which allows the INIT_CONF JTAG instruction to initiate FPGA configuration. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor. An internal pull-up on the nINIT_CONF pin is always active in the enhanced configuration devices, which means an external pull-up is not required if nCONFIG is tied to nINIT_CONF.

Upon power-up, the APEX II device goes through a Power-On Reset (POR) for approximately 5 µs. During POR, the device resets and holds nSTATUS low, and tri-states all user I/O pins. The configuration device also goes through a POR delay to allow the power supply to stabilize. The POR time for enhanced configuration devices can be set to either 100 ms or 2 ms, depending on its PORSEL pin setting. If the PORSEL pin is connected to GND, the POR delay is 100 ms. During this time, the configuration device drives its OE pin low. This low signal delays configuration because the OE pin is connected to the target device's nSTATUS pin. When both devices complete POR, they release their open-drain OE or nSTATUS pin, which is then pulled high by a pull-up resistor. Once the FPGA successfully exits POR, all user I/O pins are tri-stated. APEX II devices have weak pull-up resistors on the user I/O pins which are on before and during configuration.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the Operating Conditions table of the *APEX II Programmable Logic Device Family Data Sheet*.

When the power supplies have reached the appropriate operating voltages, the target FPGA senses the low-to-high transition on nCONFIG and initiates the configuration cycle. The configuration cycle consists of 3 stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. The beginning of configuration can be delayed by holding the nCONFIG or nSTATUS pin low.

VCCINT and VCCIO pins on the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the nSTATUS pin, which is pulled high by a pull-up resistor. Enhanced configuration devices have an optional internal pull-up on the OE pin. This option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If this internal pull-up is not used, an external 1-kΩ pull-up on the OE/nSTATUS line is required. Once nSTATUS is released the FPGA is ready to receive configuration data and the configuration stage begins.

When nSTATUS is pulled high, OE of the configuration device also goes high and the configuration device clocks data out serially to the FPGA using its internal oscillator. The APEX II device receives configuration data on its DATA[7..0] pins and the clock is received on the DCLK pin. A byte of data is latched into the FPGA on the rising edge of DCLK.

After the FPGA has received all configuration data successfully it releases the open-drain CONF_DONE pin, which is pulled high by a pull-up resistor. Since CONF_DONE is tied to the configuration device's nCS pin, the configuration device is disabled when CONF_DONE goes high. Enhanced configuration devices have an optional internal pull-up on the nCS pin. This option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If this internal pull-up is not used, an external 1kΩ pull-up on the nCS/CONF_DONE line is required. A low to high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin.

In APEX II devices, the initialization clock source is either the APEX II internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the APEX II device will take care to provide itself with enough clock cycles for proper initialization. You also have the flexibility to synchronize initialization of multiple devices by using the CLKUSR option. The *Enable user-supplied start-up clock (CLKUSR)* option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR will not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, APEX II devices require 40 clock cycles to initialize properly.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The *Enable INIT_DONE output* option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used it will be high due to an external 1-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low. When initialization is complete, the INIT_DONE pin will be released and pulled high. In user-mode, the user I/O pins will no longer have weak pull-ups and will function as assigned in your design. The enhanced configuration device will drive DCLK low and DATA[7..0] high at the end of configuration.

If an error occurs during configuration, the FPGA drives its nSTATUS pin low, resetting itself internally. Since the nSTATUS pin is tied to OE, the configuration device will also be reset. If the *Auto-Restart Configuration After Error* option-available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box-is turned on, the FPGA will automatically initiate reconfiguration if an error occurs. The APEX II device will release its nSTATUS pin after a reset time-out period (maximum of 40 µs). When the nSTATUS pin is released and pulled high by a pull-up resistor, the configuration device reconfigures the chain. If

this option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 8 µs to restart configuration. The external system can pulse nCONFIG, if nCONFIG is under system control rather than tied to V_{CC}.

In addition, if the configuration device sends all of its data and then detects that CONF_DONE has not gone high, it recognizes that the FPGA has not configured successfully. Enhanced configuration devices wait for 64 DCLK cycles after the last configuration bit was sent for CONF_DONE to reach a high state. In this case, the configuration device pulls its OE pin low, which in turn drives the target device's nSTATUS pin low. If the *Auto-Restart Configuration After Error* option is set in the software, the target device resets and then release its nSTATUS pin after a reset time-out period (maximum of 40 µs). When nSTATUS returns high, the configuration device will try to reconfigure the FPGA.

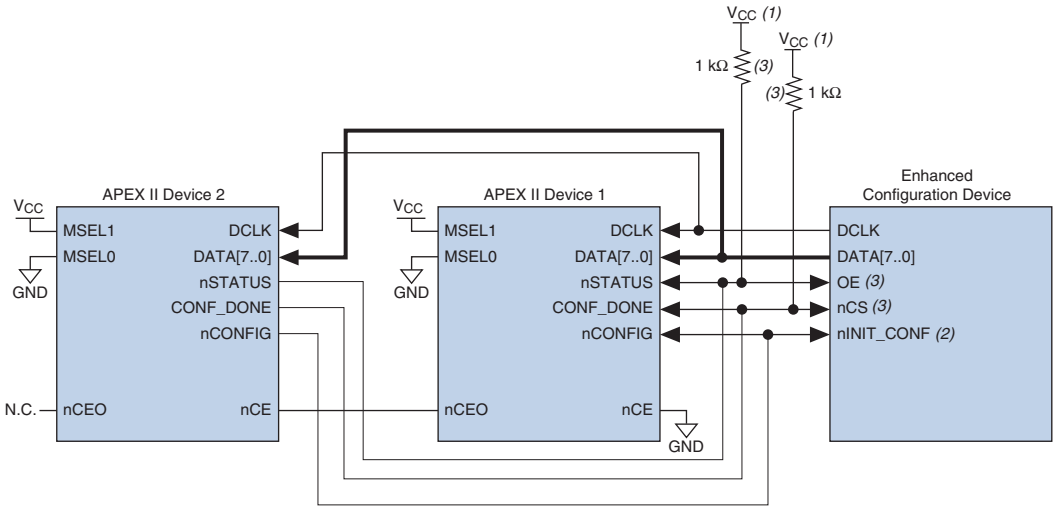
When CONF_DONE is sensed low after configuration, the configuration device recognizes that the target device has not configured successfully; therefore, your system should not pull CONF_DONE low to delay initialization. Instead, you should use the *CLKUSR* option to synchronize the initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain will initialize together if their CONF_DONE pins are tied together.

If the optional CLKUSR pin is being used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure CLKUSR continues toggling during the time nSTATUS is low (maximum of 40 µs).

When the FPGA is in user-mode, a reconfiguration can be initiated by pulling the nCONFIG pin low. The nCONFIG pin should be low for at least 8 µs. When nCONFIG is pulled low, the FPGA also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Since CONF_DONE is pulled low, this will activate the configuration device since it will see its nCS pin drive low. Once nCONFIG returns to a logic high state and nSTATUS is released by the FPGA, reconfiguration begins.

Figure 6–17 shows how to configure multiple APEX II devices with an enhanced configuration device. This circuit is similar to the configuration device circuit for a single device, except the APEX II devices are cascaded for multi-device configuration.

Figure 6–17. Multi-Device FPP Configuration Using an Enhanced Configuration Device



Notes to Figure 6–17:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active. This means an external pull-up resistor is not required on the nINIT_CONF/nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the *Disable nCS and OE pull-up resistors on configuration device* option when generating programming files.

Enhanced configuration devices (EPC4/8/16) cannot be cascaded.

When performing multi-device configuration, you must generate the configuration device's POF from each project's SOF. You can combine multiple SOFs using the Quartus II software.

For more information on how to create configuration files for multi-device configuration chains, refer to *Software Settings*, chapter 6 and 7 in volume 2 of the *Configuration Handbook*.

In multi-device FPP configuration the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin

configuration. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they may require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DCLK and DATA lines are buffered for every fourth device.

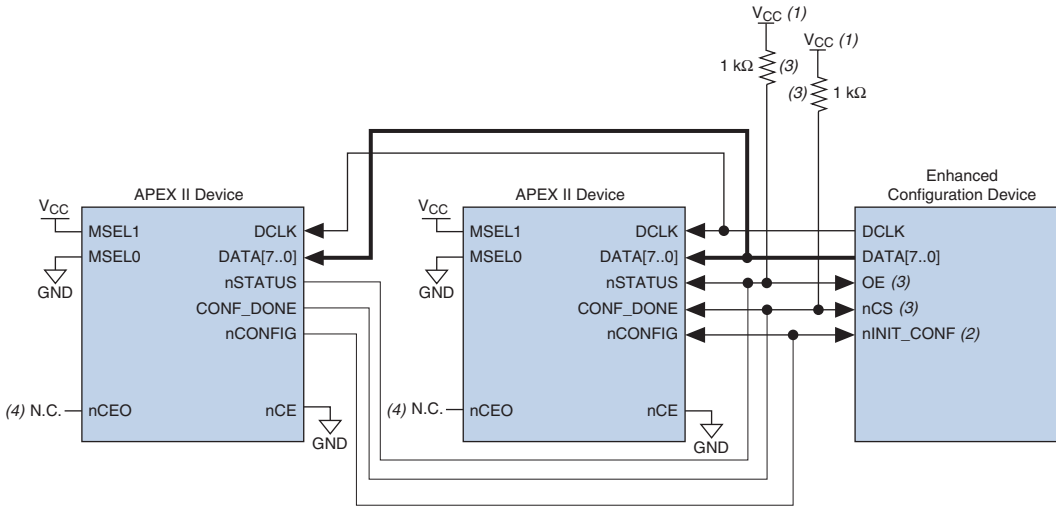
When configuring multiple devices, configuration does not begin until all devices release their OE or nSTATUS pins. Similarly, since all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Since all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first FPGA flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This low signal drives the OE pin low on the enhanced configuration device and drives nSTATUS low on all FPGAs, which causes them to enter a reset state. This behavior is similar to a single FPGA detecting an error.

If the *Auto-Restart Configuration After Error* option is turned on, the devices will automatically initiate reconfiguration if an error occurs. The FPGAs will release their nSTATUS pins after a reset time-out period (maximum of 40 µs). When all the nSTATUS pins are released and pulled high, the configuration device tries to reconfigure the chain. If the *Auto-Restart Configuration After Error* option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 8 µs to restart configuration. The external system can pulse nCONFIG if nCONFIG is under system control rather than tied to V_{CC}.

In your system, you may have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they may require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete configuration at the same time. Figure 6–18 shows multi-device FPP configuration when both APEX II devices are receiving the same configuration data.

Figure 6–18. Multiple-Device FPP Configuration Using an Enhanced Configuration Device When Both FPGAs Receive the Same Data



Notes to Figure 6–18:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active. This means an external pull-up resistor is not required on the nINIT_CONF/nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the *Disable nCS and OE pull-ups on configuration device* option when generating programming files.

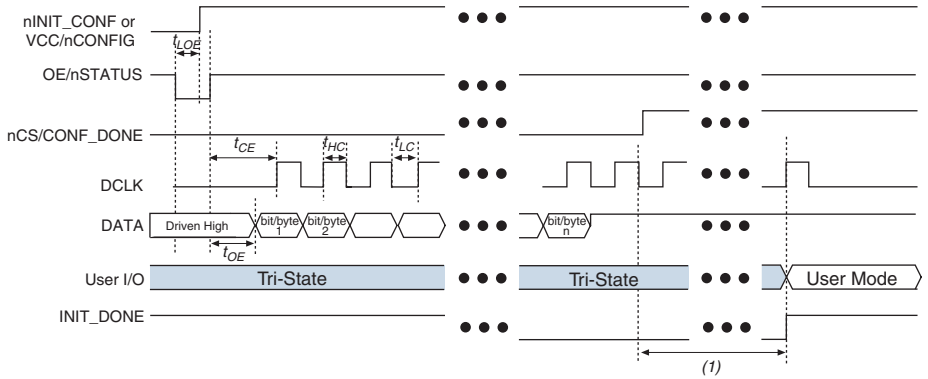
(4) The nCEO pins of both devices are left unconnected when configuring the same configuration data into multiple devices.

You can use a single enhanced configuration chain to configure multiple APEX II devices with other Altera devices that support FPP configuration, such as Stratix® and Stratix GX devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information on configuring multiple Altera devices in the same configuration chain, refer to *Configuring Mixed Altera FPGA Chains* in the *Configuration Handbook*.

Figure 6–19 shows the timing waveform for the FPP configuration scheme using an enhanced configuration device.

Figure 6–19. APEX II FPP Configuration Using an Enhanced Configuration Device Timing Waveform



Note to Figure 6–19:

(1) APEX II devices enter user mode 40 clock cycles after CONF_DONE goes high. The initialization clock can come from the APEX II internal oscillator or the CLKUSR pin.

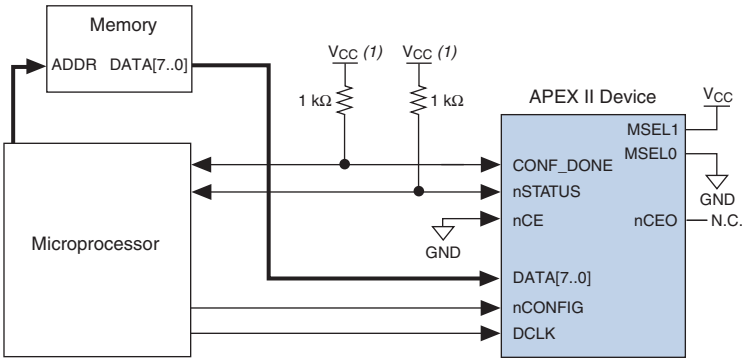
For timing information, refer to the *Enhanced Configuration Devices (EPC4, EPC8, & EPC16) Data Sheet* in the *Configuration Handbook*.

Device configuration options and how to create configuration files are discussed further in *Software Settings*, chapter 6 and 7 in volume 2 of the *Configuration Handbook*.

FPP Configuration Using a Microprocessor

In the FPP configuration scheme, an intelligent host, such as a microprocessor or CPLD, can transfer configuration data from a storage device, such as flash memory, to the target APEX II device. Configuration data can be stored in RBF, HEX or TTF format. Figure 6–20 shows the configuration interface connections between the APEX II device and a microprocessor for single device configuration.

Figure 6–20. Single Device FPP Configuration Using a Microprocessor



Note to Figure 6–20:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for the device.

Upon power-up, the APEX II device goes through a Power-On Reset (POR) for approximately 5 µs. During POR, the device resets and holds nSTATUS low, and tri-states all user I/O pins. Once the FPGA successfully exits POR, all user I/O pins are tri-stated. APEX II devices have weak pull-up resistors on the user I/O pins which are on before and during configuration.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the Operating Conditions table of the *APEX II Programmable Logic Device Family Data Sheet*.

The configuration cycle consists of three stages: reset, configuration and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration, the microprocessor must generate a low-to-high transition on the nCONFIG pin.

VCCINT and VCCIO pins on the banks where the configuration, and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 1-kΩ pull-up resistor. Once nSTATUS is released, the FPGA is ready to receive configuration data and the configuration stage begins. When nSTATUS is pulled high, the microprocessor should place the configuration data one byte at a time on the DATA[7..0] pins.

The APEX II device receives configuration data on its DATA[7..0] pins and the clock is received on the DCLK pin. Data is latched into the FPGA on the rising edge of DCLK. Data is continuously clocked into the target device until CONF_DONE goes high. After the FPGA has received all configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by an external 1-kΩ pull-up resistor. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin.

In APEX II devices, the initialization clock source is either the APEX II internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the APEX II device will take care to provide itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device. Driving DCLK to the device after configuration is complete does not affect device operation.

You also have the flexibility to synchronize initialization of multiple devices by using the CLKUSR option. The *Enable user-supplied start-up clock (CLKUSR)* option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR will not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, APEX II devices require 40 clock cycles to initialize properly.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. This *Enable INIT_DONE output* option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used it will be high due to an external 1-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low. When initialization is complete, the INIT_DONE pin will be released and pulled high. The microprocessor must be able to detect this low-to-high transition which signals the FPGA has entered user mode. In user-mode, the user I/O pins will no longer have weak pull-ups and will function as assigned in your design. When initialization is complete, the FPGA enters user mode.

To ensure DCLK and DATA0 are not left floating at the end of configuration, the microprocessor must take care to drive them either high or low, whichever is convenient on your board. The DATA[7..1] pins are available as user I/O pins after configuration. When the FPP scheme is chosen in the Quartus II software, as a default these I/O pins are tri-stated in user mode and should be driven by the microprocessor. To change this default option in the Quartus II software, select the **Dual-Purpose Pins** tab of the **Device & Pin Options** dialog box.

Handshaking signals are not used in FPP configuration mode. Therefore, the configuration clock (DCLK) speed must be below the specified frequency to ensure correct configuration. No maximum DCLK period exists, which means you can pause configuration by halting DCLK for an indefinite amount of time.

If an error occurs during configuration, the FPGA drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the microprocessor that there is an error. If the *Auto-Restart Configuration After Error* option-available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box-is turned on, the FPGA releases nSTATUS after a reset time-out period (maximum of 40 µs). After nSTATUS is released and pulled high by a pull-up resistor, the microprocessor can try to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition (with a low pulse of at least 8 µs) on nCONFIG to restart the configuration process.

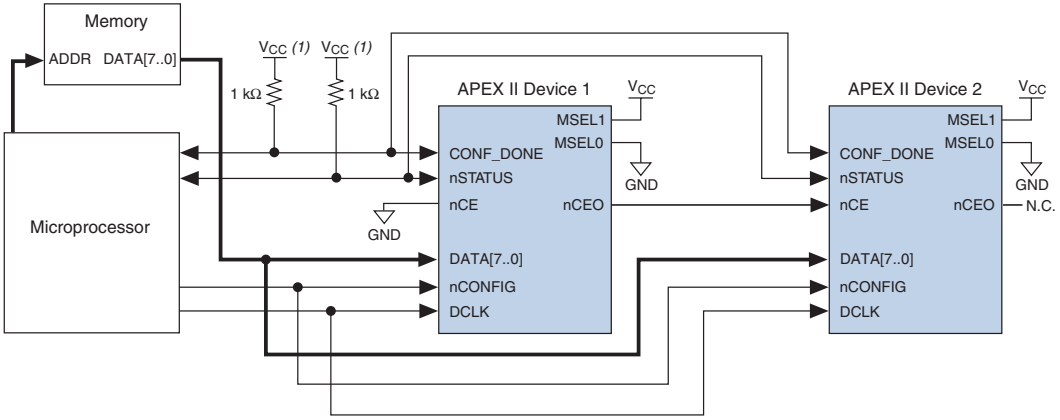
The microprocessor can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The CONF_DONE pin must be monitored by the microprocessor to detect errors and determine when programming completes. If the microprocessor sends all configuration data but CONF_DONE or INIT_DONE have not gone high, the microprocessor must reconfigure the target device.

If the optional CLKUSR pin is being used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure CLKUSR continues toggling during the time nSTATUS is low (maximum of 40 µs).

When the FPGA is in user-mode, a reconfiguration can be initiated by transitioning the nCONFIG pin low-to-high. The nCONFIG pin should be low for at least 8 µs. When nCONFIG is pulled low, the FPGA also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Once nCONFIG returns to a logic high state and nSTATUS is released by the FPGA, reconfiguration begins.

Figure 6–21 shows how to configure multiple devices using a microprocessor. This circuit is similar to the FPP configuration circuit for a single device, except the APEX II devices are cascaded for multi-device configuration.

Figure 6–21. Multi-Device FPP Configuration Using a Microprocessor



Note to Figure 6–21:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.

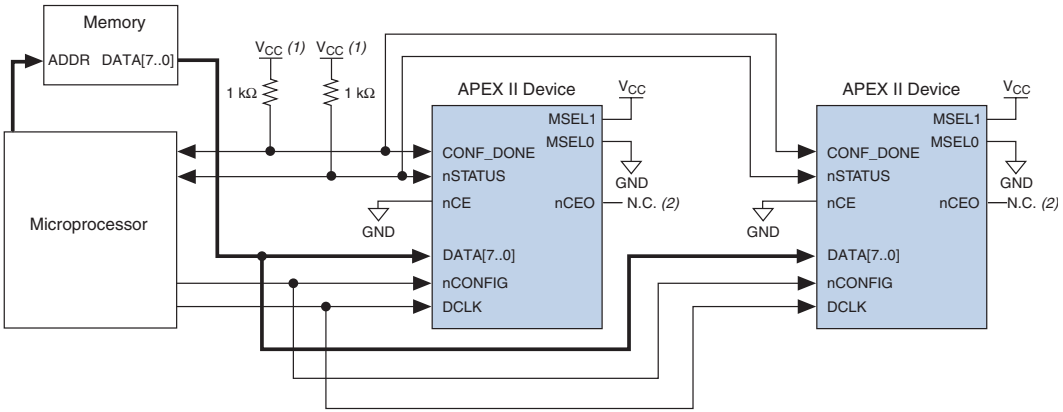
In multi-device FPP configuration the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration within one clock cycle; therefore, the transfer of data destinations is transparent to the microprocessor. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they may require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DCLK and DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Since all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first FPGA flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single FPGA detecting an error.

If the *Auto-Restart Configuration After Error* option is turned on, the FPGAs release their nSTATUS pins after a reset time-out period (maximum of 40 µs). After all nSTATUS pins are released and pulled high, the microprocessor can try to reconfigure the chain without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition (with a low pulse of at least 8 µs) on nCONFIG to restart the configuration process.

In your system, you may have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they may require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete configuration at the same time. Figure 6–22 shows multi-device FPP configuration when both APEX II devices are receiving the same configuration data.

Figure 6–22. Multiple-Device FPP Configuration Using a Microprocessor When Both FPGAs Receive the Same Data



Notes to Figure 6–22:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.

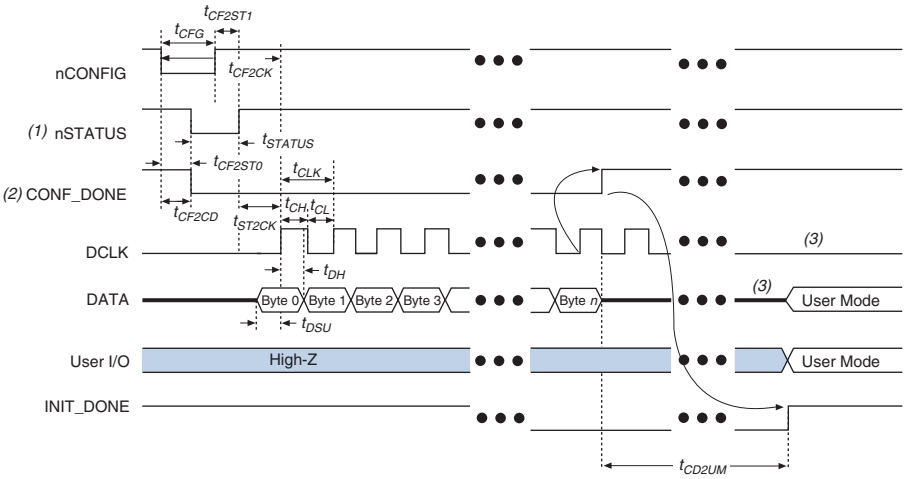
(2) The nCEO pins of both devices are left unconnected when configuring the same configuration data into multiple devices.

You can use a single configuration chain to configure APEX II devices with other Altera devices that support FPP configuration, such as Stratix. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information on configuring multiple Altera devices in the same configuration chain, refer to *Configuring Mixed Altera FPGA Chains* in the *Configuration Handbook*.

Figure 6–23 shows the timing waveform for the FPP configuration scheme using a microprocessor.

Figure 6–23. APEX II FPP Configuration Using a Microprocessor Timing Waveform



Notes to Figure 6–23:

(1) Upon power-up, the APEX II device holds nSTATUS low for not more than 5 µs after V_{CC} reaches its minimum requirement.

(2) Upon power-up, before and during configuration, CONF_DONE is low.

(3) DATA0 and DCLK should not be left floating after configuration. It should be driven high or low, whichever is more convenient. DATA[7..1] are available as user I/O pins after configuration and the state of theses pins depends on the design programmed into the device.

Table 6–5 defines the timing parameters for APEX II devices for FPP configuration.

Table 6–5. FPP Timing Parameters for APEX II Devices (Part 1 of 2) *Note (1)*

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		200	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		200	ns
t_{CFG}	nCONFIG low pulse width	8		µs
t_{STATUS}	nSTATUS low pulse width	10	40 *(2)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high		1 *(2)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	40		µs
t_{ST2CK}	nSTATUS high to first rising edge on DCLK	1		µs
t_{DSU}	Data setup time before rising edge on DCLK	10		ns
t_{DH}	Data hold time after rising edge on DCLK	0		ns

Table 6–5. FPP Timing Parameters for APEX II Devices (Part 2 of 2) Note (1)				
Symbol	**Parameter**	**Min**	**Max**	**Units**
t_{CH}	DCLK high time	7.5		ns
t_{CL}	DCLK low time	7.5		ns
t_{CLK}	DCLK period	15		ns
f_{MAX}	DCLK frequency		66	MHz
t_{CD2UM}	CONF_DONE high to user mode (3)	2	8	µs

Notes to Table 6–5:
(1) This information is preliminary.
(2) This value is obtainable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.
(3) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device. If the clock source is CLKUSR, multiply the clock period by 40 to obtain this value.

Device configuration options and how to create configuration files are discussed further in *Software Settings*, chapter 6 and 7 in volume 2 of the *Configuration Handbook*.

Configuring Using the MicroBlaster Driver

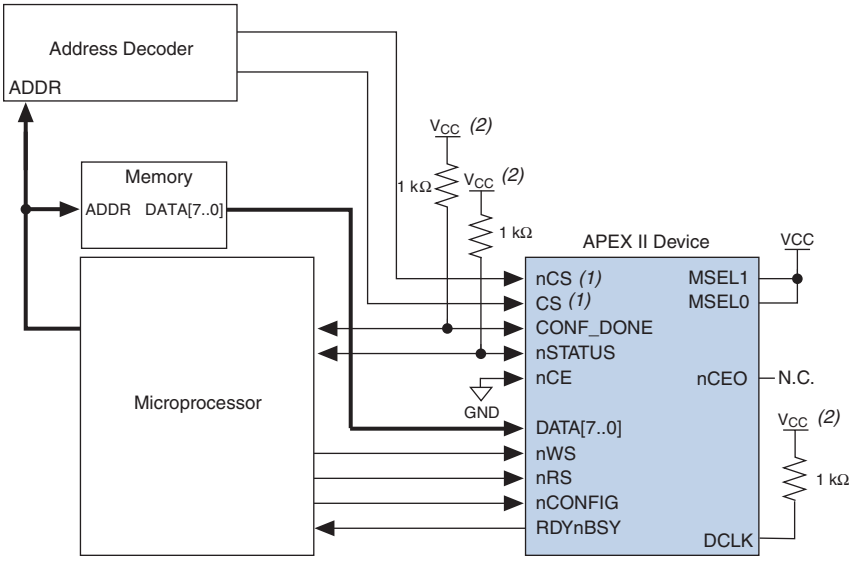
The MicroBlaster™ software driver supports a RBF programming input file and is targeted for embedded fast passive parallel configuration. The source code is developed for the Windows NT operating system, although you can customize it to run on other operating systems. For more information on the MicroBlaster software driver, go to the Altera web site (***http://www.altera.com*).**

Passive Parallel Asynchronous Configuration

Passive Parallel Asynchronous (PPA) configuration uses an intelligent host, such as a microprocessor, to transfer configuration data from a storage device, such as flash memory, to the target APEX II device. Configuration data can be stored in TTF, RBF or HEX format. The host system outputs byte-wide data and the accompanying strobe signals to the FPGA. When using PPA, you should pull the DCLK pin high through a 1-kΩ pull-up resistor to prevent unused configuration input pins from floating.

Figure 6–24 shows the configuration interface connections between the FPGA and a microprocessor for single device PPA configuration. The microprocessor or an optional address decoder can control the device's chip select pins, nCS and CS. The address decoder allows the microprocessor to select the APEX II device by accessing a particular address, which simplifies the configuration process. The nCS and CS pins must be held active during configuration and initialization.

Figure 6–24. Single Device PPA Configuration Using a Microprocessor Note (1)



Notes to Figure 6–24:

(1) If not used, the CS pin can be connected to V_{CC} directly. If not used, the nCS pin can be connected to GND directly.

(2) The pull-up resistor should be connected to a supply that provides an acceptable input signal for the device.

During PPA configuration, it is only required to use either the nCS or CS pin. Therefore, if only one chip-select input is used, the other must be tied to the active state. For example, nCS can be tied to ground while CS is toggled to control configuration. The device's nCS or CS pins can be toggled during PPA configuration if the design meets the specifications set for t_{CSSU}, t_{WSP}, and t_{CSH} listed in Table 6–6.

Upon power-up, the APEX II device goes through a Power-On Reset (POR) for approximately 5 µs. During POR, the device resets and holds nSTATUS low, and tri-states all user I/O pins. Once the FPGA successfully exits POR, all user I/O pins are tri-stated. APEX II devices have weak pull-up resistors on the user I/O pins which are on before and during configuration.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the Operating Conditions table of the *APEX II Programmable Logic Device Family Data Sheet*.

The configuration cycle consists of three stages: reset, configuration and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration, the microprocessor must generate a low-to-high transition on the nCONFIG pin.

VCCINT and VCCIO pins on the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 1-kΩ pull-up resistor. Once nSTATUS is released the FPGA is ready to receive configuration data and the configuration stage begins. When nSTATUS is pulled high, the microprocessor should then assert the target device's nCS pin low and/or CS pin high. Next, the microprocessor places an 8-bit configuration word (one byte) on the target device's DATA[7..0] pins and pulses the nWS pin low.

On the rising edge of nWS, the target device latches in a byte of configuration data and drives its RDYnBSY signal low, which indicates it is processing the byte of configuration data. The microprocessor can then perform other system functions while the APEX II device is processing the byte of configuration data.

During the time RDYnBSY is low, the APEX II device internally processes the configuration data using its internal oscillator (typically 10 MHz). When the device is ready for the next byte of configuration data, it will drive RDYnBSY high. If the microprocessor senses a high signal when it polls RDYnBSY, the microprocessor sends the next byte of configuration data to the FPGA.

Alternatively, the nRS signal can be strobed low, causing the RDYnBSY signal to appear on DATA7. Because RDYnBSY does not need to be monitored, this pin doesn't need to be connected to the microprocessor. Data should not be driven onto the data bus while nRS is low because it will cause contention on the DATA7 pin. If the nRS pin is not used to monitor configuration, it should be tied high.

To simplify configuration and save an I/O port, the microprocessor can wait for the total time of t_{BUSY}(max) + t_{RDY2WS} + t_{W2SB} before sending the next data byte. In this set-up, nRS should be tied high and RDYnBSY does not need to be connected to the microprocessor. The t_{BUSY}, t_{RDY2WS} and t_{W2SB} timing specifications are listed in Table 6–6.

Next, the microprocessor checks nSTATUS and CONF_DONE. If nSTATUS is not low and CONF_DONE is not high, the microprocessor sends the next data byte. However, if nSTATUS is not low and all the configuration data has been received, the device is ready for initialization. After the FPGA has received all configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by an external 1-kΩ pull-up resistor. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin.

In APEX II devices, the initialization clock source is either the APEX II internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the APEX II device will take care to provide itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device.

You also have the flexibility to synchronize initialization of multiple devices by using the CLKUSR option. The *Enable user-supplied start-up clock (CLKUSR)* option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR will not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, APEX II devices require 40 clock cycles to initialize properly.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. This *Enable INIT_DONE output* option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used it will be high due to an external 1-kΩ pull-up when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low. When initialization is complete, the INIT_DONE pin will be released and pulled high. The microprocessor must be able to detect this low-to-high transition which signals the FPGA has entered user mode. In user-mode, the user I/O pins will no longer have weak pull-ups and will function as assigned in your design. When initialization is complete, the FPGA enters user mode.

To ensure DATA0 is not left floating at the end of configuration, the microprocessor must take care to drive them either high or low, whichever is convenient on your board. After configuration, the nCS, CS, nRS, nWS, RDYnBSY, and DATA[7..1] pins can be used as user I/O pins. When the PPA scheme is chosen in the Quartus II software, as a default

these I/O pins are tri-stated in user mode and should be driven by the microprocessor. To change this default option in the Quartus II software, select the **Dual-Purpose Pins** tab of the **Device & Pin Options** dialog box.

If an error occurs during configuration, the FPGA drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the microprocessor that there is an error. If the *Auto-Restart Configuration After Error* option-available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box-is turned on, the FPGA releases nSTATUS after a reset time-out period (maximum of 40 µs). After nSTATUS is released and pulled high by a pull-up resistor, the microprocessor can try to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition (with a low pulse of at least 8 µs) on nCONFIG to restart the configuration process.

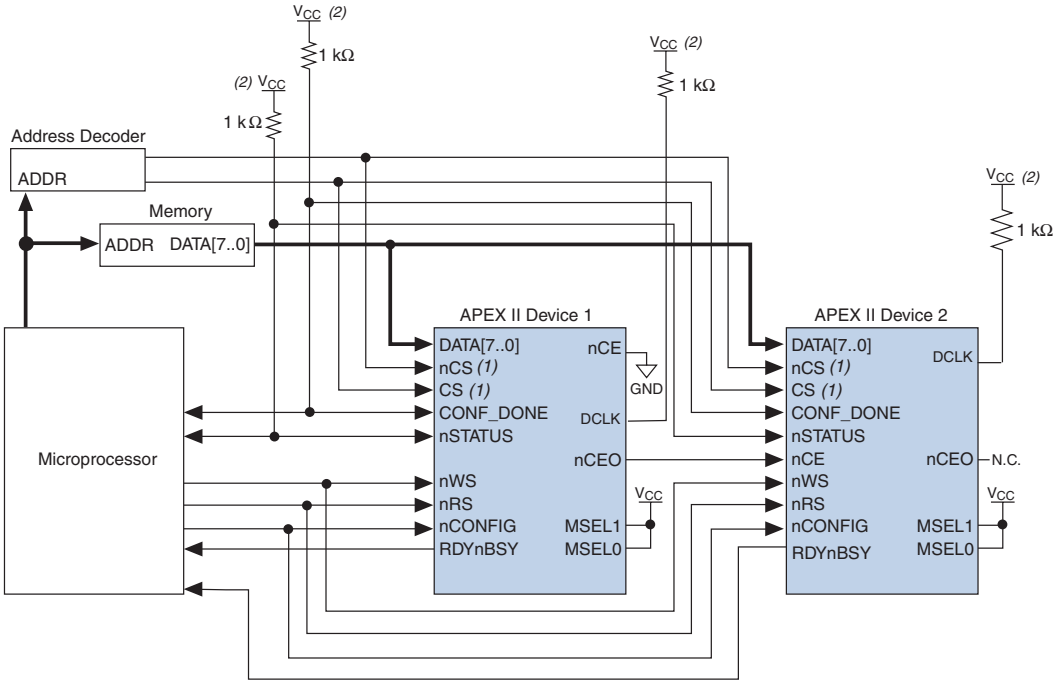
The microprocessor can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The CONF_DONE pin must be monitored by the microprocessor to detect errors and determine when programming completes. If the microprocessor sends all configuration data but CONF_DONE or INIT_DONE has not gone high, the microprocessor must reconfigure the target device.

If the optional CLKUSR pin is being used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure CLKUSR continues toggling during the time nSTATUS is low (maximum of 40 µs).

When the FPGA is in user-mode, a reconfiguration can be initiated by transitioning the nCONFIG pin low-to-high. The nCONFIG pin should be low for at least 8 µs. When nCONFIG is pulled low, the FPGA also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Once nCONFIG returns to a logic high state and nSTATUS is released by the FPGA, reconfiguration begins.

Figure 6–25 shows how to configure multiple APEX II devices using a microprocessor. This circuit is similar to the PPA configuration circuit for a single device, except the devices are cascaded for multi-device configuration.

Figure 6–25. Multi-Device PPA Configuration Using a Microprocessor



Notes to Figure 6–25:

(1) If not used, the CS pin can be connected to V_{CC} directly. If not used, the nCS pin can be connected to GND directly.

(2) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.

In multi-device PPA configuration the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration within one clock cycle; therefore, the transfer of data destinations is transparent to the microprocessor.

Each device's RDYnBSY pin can have a separate input to the microprocessor. Alternatively, if the microprocessor is pin limited, all the RDYnBSY pins can feed an AND gate and the output of the AND gate can feed the microprocessor. For example, if you have 2 devices in a PPA configuration chain, the second device's RDYnBSY pin will be high during the time that the first device is being configured. When the first device has

been successfully configured, it will driven nCEO low to activate the next device in the chain and drive its RDYnBSY pin high. Therefore, since RDYnBSY signal is driven high before configuration and after configuration before entering user-mode, the device being configured will govern the output of the AND gate.

The nRS signal can be used in multi-device PPA chain since the APEX II device will tri-state its DATA[7..0] pins before configuration and after configuration before entering user-mode to avoid contention. Therefore, only the device that is currently being configured will respond to the nRS strobe by asserting DATA7.

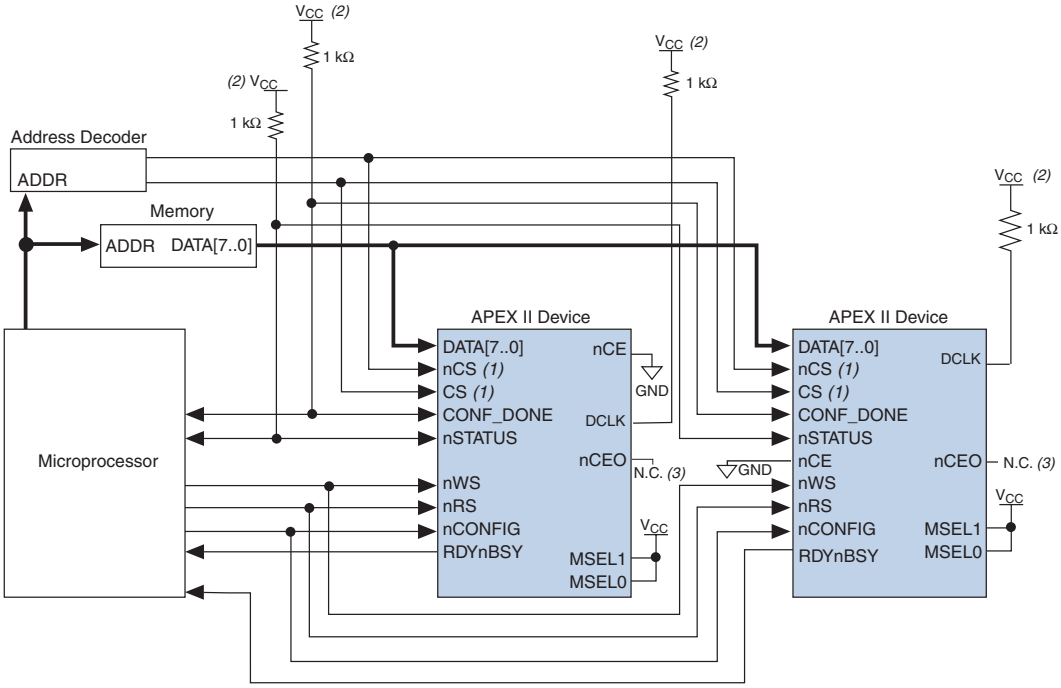
All other configuration pins (nCONFIG, nSTATUS, DATA[7..0], nCS, CS, nWS, nRS and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they may require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Since all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first FPGA flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single FPGA detecting an error.

If the *Auto-Restart Configuration After Error* option is turned on, the FPGAs release their nSTATUS pins after a reset time-out period (maximum of 40 µs). After all nSTATUS pins are released and pulled high, the microprocessor can try to reconfigure the chain without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition (with a low pulse of at least 8 µs) on nCONFIG to restart the configuration process.

In your system, you may have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DATA[7..1], nCS, CS, nWS, nRS and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they may require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete configuration at the same time. Figure 6–26 shows multi-device PPA configuration when both devices are receiving the same configuration data.

Figure 6–26. Multiple-Device PPA Configuration Using a Microprocessor When Both FPGAs Receive the Same Data



Notes to Figure 6–26:

(1) If not used, the CS pin can be connected to V_{CC} directly. If not used, the nCS pin can be connected to GND directly.

(2) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.

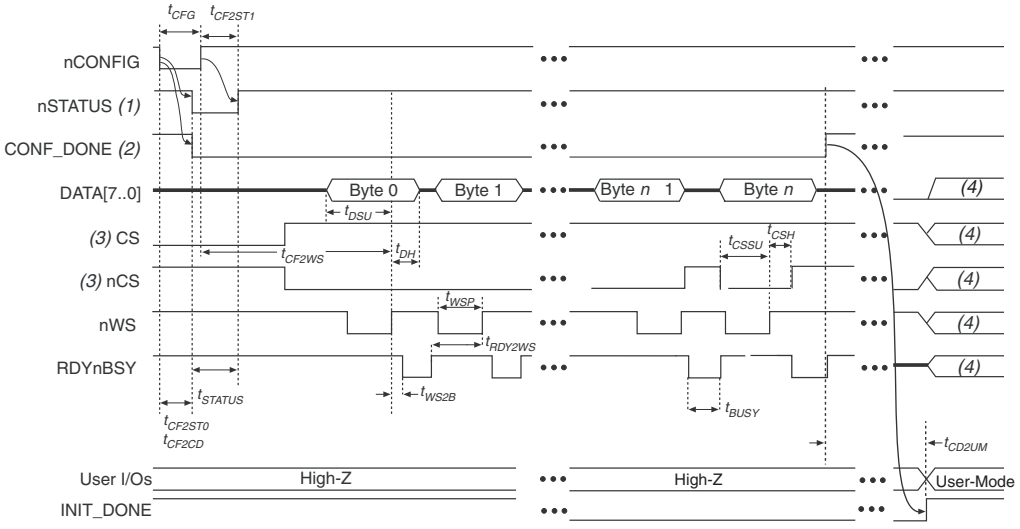
(3) The nCEO pins of both devices are left unconnected when configuring the same configuration data into multiple devices.

You can use a single configuration chain to configure APEX II devices with other Altera devices that support PPA configuration, such as Stratix, Mercury, APEX 20K, ACEX 1K, and FLEX 10KE devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information on configuring multiple Altera devices in the same configuration chain, refer to *Configuring Mixed Altera FPGA Chains* in the *Configuration Handbook*.

Figure 6–27 shows the timing waveform for the PPA configuration scheme using a microprocessor.

Figure 6–27. APEX II PPA Configuration Timing Waveform



Notes to Figure 6–27:

(1) Upon power-up, the APEX II device holds nSTATUS low for not more than 5 µs after V_{CCINT} reaches its minimum requirement.

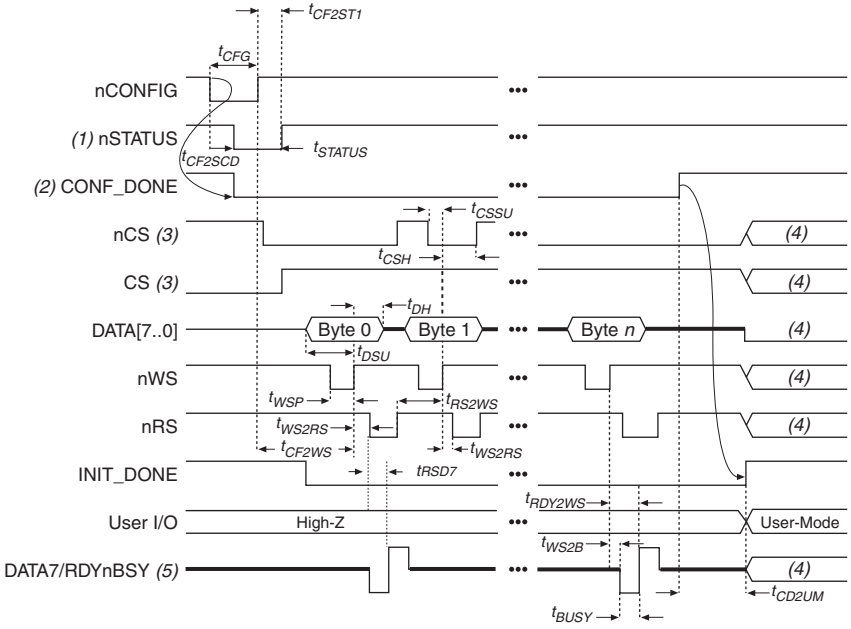
(2) Upon power-up, before and during configuration, CONF_DONE is low.

(3) The user can toggle nCS or CS during configuration if the design meets the specification for t_{CSSU}, t_{WSP}, and t_{CSH}.

(4) DATA0 should not be left floating after configuration. It should be driven high or low, whichever is more convenient. DATA[7..1], CS, nCS, nWS, nRS and RDYnBSY are available as user I/O pins after configuration and the state of theses pins depends on the dual-purpose pin settings.

Figure 6–28 shows the timing waveform for the PPA configuration scheme when using a strobed nRS and nWS signal.

Figure 6–28. APEX II PPA Configuration Timing Waveform Using nRS & nWS



Notes to Figure 6–28:

(1) Upon power-up, the APEX II device holds nSTATUS low for not more than 5 µs after V_{CCINT} reaches its minimum requirement.

(2) Upon power-up, before and during configuration, CONF_DONE is low.

(3) The user can toggle nCS or CS during configuration if the design meets the specification for t_{CSSU}, t_{WSP}, and t_{CSH}.

(4) DATA0 should not be left floating after configuration. It should be driven high or low, whichever is more convenient. DATA[7..1], CS, nCS, nWS, nRS, and RDYnBSY are available as user I/O pins after configuration and the state of theses pins depends on the dual-purpose pin settings.

(5) DATA7 is a bidirectional pin. It is an input for configuration data input, but it is an output to show the status of RDYnBSY.

Table 6–6 defines the timing parameters for APEX II devices for PPA configuration.

Table 6–6. PPA Timing Parameters for APEX II Devices (Part 1 of 2) *Note (1)*

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		200	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		200	ns
t_{CFG}	nCONFIG low pulse width	8		µs
t_{STATUS}	nSTATUS low pulse width	10	40 *(2)*	µs

Table 6–6. PPA Timing Parameters for APEX II Devices (Part 2 of 2) *Note (1)*

Symbol	Parameter	Min	Max	Units
t_{CF2ST1}	nCONFIG high to nSTATUS high		1 *(2)*	µs
t_{CSSU}	Chip select setup time before rising edge on nWS	10		ns
t_{CSH}	Chip select hold time after rising edge on nWS	0		ns
t_{CF2WS}	nCONFIG high to first rising edge on nWS	40		µs
t_{DSU}	Data setup time before rising edge on nWS	10		ns
t_{DH}	Data hold time after rising edge on nWS	0		ns
t_{WSP}	nWS low pulse width	200		ns
t_{WS2B}	nWS rising edge to RDYnBSY low		50	ns
t_{BUSY}	RDYnBSY low pulse width	0.1	1.6	µs
t_{RDY2WS}	RDYnBSY rising edge to nWS rising edge	50		ns
t_{WS2RS}	nWS rising edge to nRS falling edge	200		ns
t_{RS2WS}	nRS rising edge to nWS rising edge	200		ns
t_{RSD7}	nRS falling edge to DATA7 valid with RDYnBSY signal		50	ns
t_{CD2UM}	CONF_DONE high to user mode *(3)*	2	8	µs

Notes to Table 6–6:

(1) This information is preliminary.

(2) This value is obtainable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(3) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device. If the clock source is CLKUSR, multiply the clock period by 40 for APEX II devices to obtain this value.

Device configuration options and how to create configuration files are discussed further in *Software Settings*, chapter 6 and 7 in volume 2 of the *Configuration Handbook*.

JTAG Configuration

The Joint Test Action Group (JTAG) has developed a specification for boundary-scan testing. This boundary-scan test (BST) architecture offers the capability to efficiently test components on PCBs with tight lead spacing. The BST architecture can test pin connections without using physical test probes and capture functional data while a device is operating normally. The JTAG circuitry can also be used to shift configuration data into the device.

For more information on JTAG boundary-scan testing, refer to *Application Note 39: IEEE 1149.1 (JTAG) Boundary-Scan Testing* in Altera Devices.

A device operating in JTAG mode uses four required pins, TDI, TDO, TMS, and TCK, and one optional pin, TRST. All user I/O pins are tri-stated during JTAG configuration. APEX II devices are designed such that JTAG instructions have precedence over any device configuration modes. This means that JTAG configuration can take place without waiting for other configuration modes to complete. For example, if you attempt JTAG configuration of APEX II FPGAs during PS configuration, PS configuration will be terminated and JTAG configuration will begin.

Table 6–7 explains each JTAG pin's function.

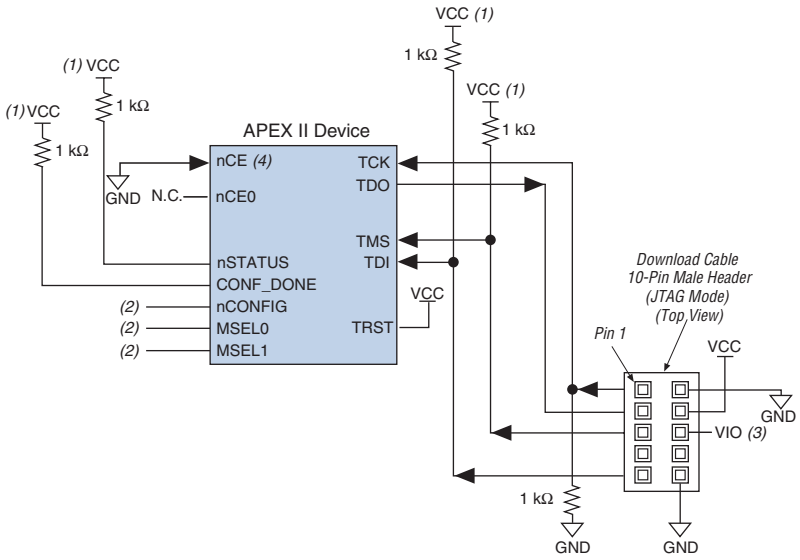
Table 6–7. JTAG Pin Descriptions

Pin	Description	Function
TDI	Test data input	Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}.
TDO	Test data output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by leaving this pin unconnected.
TMS	Test mode select	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}.
TCK	Test clock input	The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND.
TRST	Test reset input (optional)	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to IEEE Std. 1149.1. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND.

If V_{CCIO} of the bank where the JTAG pins reside, are tied to 3.3-V, both the I/O pins and JTAG TDO port will drive at 3.3-V levels.

During JTAG configuration, data can be downloaded to the device on the PCB through the USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV header. Configuring devices through a cable is similar to programming devices in-system, except the TRST pin should be connected to V_{CC}. This ensures that the TAP controller is not reset. Figure 6–29. shows JTAG configuration of a single APEX II device.

Figure 6–29. JTAG Configuration of a Single Device Using a Download Cable



Notes to Figure 6–29:

(1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster (VIO pin), ByteBlaster II, or ByteBlasterMV cable.

(2) The nCONFIG, MSEL0, and MSEL1 pins should be connected to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect nCONFIG to V_{CC}, and MSEL0 and MSEL1 to ground.

(3) Pin 6 of the header is a VIO reference voltage for the MasterBlaster output driver. VIO should match the device's V_{CCIO}. Refer to the MasterBlaster Serial/USB Communications Cable Data Sheet for this value. In the ByteBlasterMV, this pin is a no connect. In the USB Blaster and ByteBlaster II, this pin is connected to nCE when it is used for Active Serial programming, otherwise it is a no connect.

(4) nCE must be connected to GND or driven low for successful JTAG configuration.

To configure a single device in a JTAG chain, the programming software places all other devices in BYPASS mode. In BYPASS mode, devices pass programming data from the TDI pin to the TDO pin through a single bypass register without being affected internally. This scheme enables the programming software to program or verify the target device. Configuration data driven into the device appears on the TDO pin one clock cycle later.

APEX II devices have dedicated JTAG pins that always function as JTAG pins. JTAG testing can be performed on APEX II devices both before and after configuration, but not during configuration. The chip-wide reset (DEV_CLRn) and chip-wide output enable (DEV_OE) pins on APEX II devices do not affect JTAG boundary-scan or programming operations. Toggling these pins does not affect JTAG operations (other than the usual boundary-scan operation).

When designing a board for JTAG configuration of APEX II devices, the dedicated configuration pins should be considered. Table 6–8 shows how these pins should be connected during JTAG configuration.

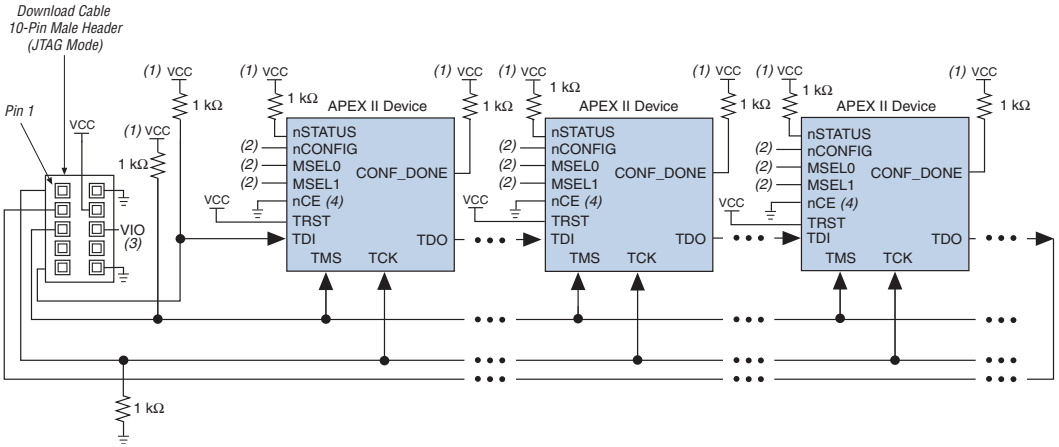
Table 6–8. Dedicated Configuration Pin Connections During JTAG Configuration

Signal	Description
nCE	On all APEX II devices in the chain, nCE should be driven low by connecting it to ground, pulling it low via a resistor, or driving it by some control circuitry. For devices that are also in multi-device PS, FPP, or PPA configuration chains, the nCE pins should be connected to GND during JTAG configuration or JTAG configured in the same order as the configuration chain.
nCEO	On all APEX II devices in the chain, nCEO can be left floating or connected to the nCE of the next device. See nCE description above.
MSEL	These pins must not be left floating. These pins support whichever non-JTAG configuration is used in production. If only JTAG configuration is used, you should tie both pins to ground.
nCONFIG	Driven high by connecting to V_{CC}, pulling high via a resistor, or driven by some control circuitry.
nSTATUS	Pull to V_{CC} via a 1-kΩ resistor. When configuring multiple devices in the same JTAG chain, each nSTATUS pin should be pulled up to V_{CC} individually. nSTATUS pulling low in the middle of JTAG configuration indicates that an error has occurred.
CONF_DONE	Pull to V_{CC} via a 1-kΩ resistor. When configuring multiple devices in the same JTAG chain, each CONF_DONE pin should be pulled up to V_{CC} individually. CONF_DONE going high at the end of JTAG configuration indicates successful configuration.
DCLK	Should not be left floating. Drive low or high, whichever is more convenient on your board.
DATA0	Should not be left floating. Drive low or high, whichever is more convenient on your board.

When programming a JTAG device chain, one JTAG-compatible header is connected to several devices. The number of devices in the JTAG chain is limited only by the drive capability of the download cable. When four or more devices are connected in a JTAG chain, Altera recommends buffering the TCK, TDI, and TMS pins with an on-board buffer.

JTAG-chain device programming is ideal when the system contains multiple devices, or when testing your system using JTAG BST circuitry. Figure 6–30 shows multi-device JTAG configuration.

Figure 6–30. JTAG Configuration of Multiple Devices Using a Download Cable



Notes to Figure 6–30:

(1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster (VIO pin), ByteBlaster II, or ByteBlasterMV cable.

(2) The nCONFIG, MSEL0, and MSEL1 pins should be connected to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect nCONFIG to V_{CC}, and MSEL0 and MSEL1 to ground.

(3) Pin 6 of the header is a VIO reference voltage for the MasterBlaster output driver. VIO should match the device's V_{CCIO}. Refer to the MasterBlaster Serial/USB Communications Cable Data Sheet for this value. In the ByteBlasterMV, this pin is a no connect. In the USB Blaster and ByteBlaster II, this pin is connected to nCE when it is used for Active Serial programming, otherwise it is a no connect.

(4) nCE must be connected to GND or driven low for successful JTAG configuration.

The nCE pin must be connected to GND or driven low during JTAG configuration. In multi-device PS, FPP, and PPA configuration chains, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. Therefore, if these devices are also in a JTAG chain, you should make sure the nCE pins are connected to GND during JTAG configuration or that the devices are JTAG configured in the same order as the configuration chain. As long as the devices are JTAG configured in the same order as the multi-device configuration chain, the nCEO of the previous device will drive nCE of the next device low when it has successfully been JTAG configured.

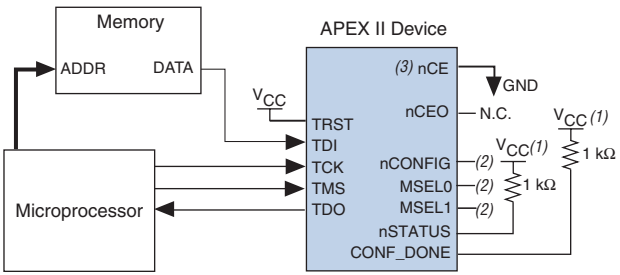
Other Altera devices that have JTAG support can be placed in the same JTAG chain for device programming and configuration.

For more information about configuring multiple Altera devices in the same configuration chain, refer to *Configuring Mixed Altera FPGA Chains* in the *Configuration Handbook*.

The Quartus II software verifies successful JTAG configuration upon completion. At the end of configuration, the software checks the state of CONF_DONE through the JTAG port. If CONF_DONE is not high, the Quartus II software indicates that configuration has failed. If CONF_DONE is high, the software indicates that configuration was successful. When Quartus II generates a JAM file for a multi-device chain, it contains instructions so that all the devices in the chain will be initialized at the same time.

Figure 6–31 shows JTAG configuration of an APEX II FPGA with a microprocessor.

Figure 6–31. JTAG Configuration of a Single Device Using a Microprocessor



Notes to Figure 6–31:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.

(2) Connect the nCONFIG, MSEL1, and MSEL0 pins to support a non-JTAG configuration scheme. If your design only uses JTAG configuration, connect the nCONFIG pin to V_{CC} and the MSEL1 and MSEL0 pins to ground.

(3) nCE must be connected to GND or driven low for successful JTAG configuration.

Jam STAPL

Jam STAPL, JEDEC standard JESD-71, is a standard file format for in-system programmability (ISP) purposes. Jam STAPL supports programming or configuration of programmable devices and testing of electronic systems, using the IEEE 1149.1 JTAG interface. Jam STAPL is a freely licensed open standard.

The Jam Player provides an interface for manipulating the IEEE Std. 1149.1 JTAG TAP state machine.

For more information on JTAG and Jam STAPL in embedded environments, refer to *AN 122: Using Jam STAPL for ISP & ICR via an Embedded Processor*. To download the jam player, visit the Altera web site:

www.altera.com/support/software/download/programming/jam/jam-index.jsp

Configuring APEX II FPGAs with JRunner

JRunner is a software driver that allows you to configure Altera FPGAs, including APEX II FPGAs, through the ByteBlaster II or ByteBlasterMV cables in JTAG mode. The programming input file supported is in RBF format. JRunner also requires a Chain Description File (**.cdf**) generated by the Quartus II software. JRunner is targeted for embedded JTAG configuration. The source code has been developed for the Windows NT operating system (OS). You can customize the code to make it run on other platforms.

For more information on the JRunner software driver, refer to the *JRunner Software Driver: An Embedded Solution to the JTAG Configuration White Paper* and the source files.

Device Configuration Pins

The following tables describe the connections and functionality of all the configuration related pins on the APEX II device. Table 6–9 describes the dedicated configuration pins, which are required to be connected properly on your board for successful configuration. Some of these pins may not be required for your configuration schemes.

Table 6–9. Dedicated Configuration Pins on the APEX II Device (Part 1 of 5)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
MSEL0 MSEL1	N/A	All	Input	Two-bit configuration input that sets the APEX II device configuration scheme. See Table 6–3 for the appropriate connections. These pins must remain at a valid state during power-up, before `nCONFIG` is pulled low to initiate a reconfiguration and during configuration.
VCCSEL	N/A	All	Input	Dedicated input that ensures the configuration related I/O banks have powered up to the appropriate 1.8-V or 2.5-V/3.3-V voltage levels before starting configuration. A logic high (1.5 V, 1.8 V, 2.5 V, 3.3 V) means 1.8 V, and a logic low means 2.5-V/3.3 V.

Table 6–9. Dedicated Configuration Pins on the APEX II Device (Part 2 of 5)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nCONFIG	N/A	All	Input	Configuration control input. Pulling this pin low during user-mode will cause the FPGA to lose its configuration data, enter a reset state, tri-state all I/O pins, and returning this pin to a logic high level will initiate a reconfiguration. If your configuration scheme uses an enhanced configuration device or EPC2 device, nCONFIG can be tied directly to V_{CC} or to the configuration device's nINIT_CONF pin.
nSTATUS	N/A	All	Bidirectional open-drain	The FPGA drives nSTATUS low immediately after power-up and releases it within 5 µs. (When using a configuration device, the configuration device holds nSTATUS low for up to 200 ms.) Status output. If an error occurs during configuration, nSTATUS is pulled low by the target device. Status input. If an external source drives the nSTATUS pin low during configuration or initialization, the target device enters an error state. Driving nSTATUS low after configuration and initialization does not affect the configured device. If a configuration device is used, driving nSTATUS low will cause the configuration device to attempt to configure the FPGA, but since the FPGA ignores transitions on nSTATUS in user-mode, the FPGA will not reconfigure. To initiate a reconfiguration, nCONFIG must be pulled low. The enhanced configuration devices' and EPC2 devices' OE and nCS pins have optional internal programmable pull-up resistors. If internal pull-up resistors are used, external 1-kΩ pull-up resistors should not be used on these pins.
CONF_DONE	N/A	All	Bidirectional open-drain	Status output. The target FPGA drives the CONF_DONE pin low before and during configuration. Once all configuration data is received without error and the initialization cycle starts, the target device releases CONF_DONE. Status input. After all data is received and CONF_DONE goes high, the target device initializes and enters user mode. Driving CONF_DONE low after configuration and initialization does not affect the configured device. The enhanced configuration devices' and EPC2 devices' OE and nCS pins have optional internal programmable pull-up resistors. If internal pull-up resistors are used, external 1-kΩ pull-up resistors should not be used on these pins.

Table 6–9. Dedicated Configuration Pins on the APEX II Device (Part 3 of 5)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nCE	N/A	All	Input	Active-low chip enable. The nCE pin activates the device with a low signal to allow configuration. The nCE pin must be held low during configuration, initialization, and user mode. In single device configuration, it should be tied low. In multi-device configuration, nCE of the first device is tied low while its nCEO pin is connected to nCE of the next device in the chain. The nCE pin must also be held low for successful JTAG programming of the FPGA.
nCEO	N/A	All	Output	Output that drives low when device configuration is complete. In single device configuration, this pin is left floating. In multi-device configuration, this pin feeds the next device's nCE pin. The nCEO of the last device in the chain is left floating.
DCLK	N/A	Synchronous configuration schemes (PS and FPP)	Input	Clock input used to clock data from an external source into the target device. Data is latched into the FPGA on the rising edge of DCLK. In PPA mode, DCLK should be tied high to V_{CC} to prevent this pin from floating. After configuration, this pin is tri-stated. In schemes that use a configuration device, DCLK will be driven low after configuration is done. In schemes that use a control host, DCLK should be driven either high or low, whichever is more convenient. Toggling this pin after configuration does not affect the configured device.
DATA0	N/A	All	Input	Data input. In serial configuration modes, bit-wide configuration data is presented to the target device on the DATA0 pin. After configuration, EPC1 and EPC1441 devices tri-state this pin, while EPC2 devices drive this pin high. In schemes that use a control host, DATA0 should be driven either high or low, whichever is more convenient. Toggling this pin after configuration does not affect the configured device.
DATA[7..1]	I/O	Parallel configuration schemes (FPP and PPA)	Inputs	Data inputs. Byte-wide configuration data is presented to the target device on DATA[7..0]. In serial configuration schemes, they function as user I/O pins during configuration, which means they are tri-stated. After PPA or FPP configuration, DATA[7..1] are available as a user I/O pins and the state of these pin depends on the Dual-Purpose Pin settings.

Table 6–9. Dedicated Configuration Pins on the APEX II Device (Part 4 of 5)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
DATA7	I/O	PPA	Bidirectional	In the PPA configuration scheme, the DATA7 pin presents the RDYnBSY signal after the nRS signal has been strobed low. In serial configuration schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, DATA7 is available as a user I/O and the state of this pin depends on the Dual-Purpose Pin settings.
nWS	I/O	PPA	Input	Write strobe input. A low-to-high transition causes the device to latch a byte of data on the DATA[7..0] pins. In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, nWS is available as a user I/O and the state of this pin depends on the Dual-Purpose Pin settings.
nRS	I/O	PPA	Input	Read strobe input. A low input directs the device to drive the RDYnBSY signal on the DATA7 pin. If the nRS pin is not used in PPA mode, it should be tied high. In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, nRS is available as a user I/O and the state of this pin depends on the Dual-Purpose Pin settings.

Table 6–9. Dedicated Configuration Pins on the APEX II Device (Part 5 of 5)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
RDYnBSY	I/O	PPA	Output	Ready output. A high output indicates that the target device is ready to accept another data byte. A low output indicates that the target device is busy and not ready to receive another data byte. In PPA configuration schemes, this pin will drive out high after power-up, before configuration and after configuration before entering user-mode. In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, RDYnBSY is available as a user I/O and the state of this pin depends on the dual-purpose pin settings.
nCS/CS	I/O	PPA	Input	Chip-select inputs. A low on nCS and a high on CS select the target device for configuration. The nCS and CS pins must be held active during configuration and initialization. During the PPA configuration mode, it is only required to use either the nCS or CS pin. Therefore, if only one chip-select input is used, the other must be tied to the active state. For example, nCS can be tied to ground while CS is toggled to control configuration. In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, nCS and CS are available as a user I/O pins and the state of these pins depends on the dual-purpose pin settings.

Table 6–10 describes the optional configuration pins. If these optional configuration pins are not enabled in the Quartus II software, they are available as general-purpose user I/O pins. Therefore during configuration they function as user I/O pins, which means they are tri-stated with weak pull-up resistors.

Table 6–10. Optional Configuration Pins

Pin Name	User Mode	Pin Type	Description
CLKUSR	N/A if option is on. I/O if option is off.	Input	Optional user-supplied clock input. Synchronizes the initialization of one or more devices. This pin is enabled by turning on the *Enable user-supplied start-up clock (CLKUSR)* option in the Quartus II software
INIT_DONE	N/A if option is on. I/O if option is off.	Output open-drain	Status pin. Can be used to indicate when the device has initialized and is in user mode. When nCONFIG is low and during the beginning of configuration, the INIT_DONE pin is tri-stated and pulled high due to an external 1-kΩ pull-up resistor. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low. When initialization is complete, the INIT_DONE pin will be released and pulled high and the FPGA enters user mode. Thus, the monitoring circuitry must be able to detect a low-to-high transition. This pin is enabled by turning on the *Enable INIT_DONE output* option in the Quartus II software.
DEV_OE	N/A if option is on. I/O if option is off.	Input	Optional pin that allows the user to override all tri-states on the device. When this pin is driven low, all I/O pins are tri-stated; when this pin is driven high, all I/O pins behave as programmed. This pin is enabled by turning on the *Enable device-wide output enable (DEV_OE)* option in the Quartus II software.
DEV_CLRn	N/A if option is on. I/O if option is off.	Input	Optional pin that allows you to override all clears on all device registers. When this pin is driven low, all registers are cleared; when this pin is driven high, all registers behave as programmed. This pin is enabled by turning on the *Enable device-wide reset (DEV_CLRn)* option in the Quartus II software.

JTAG pins must be kept stable before and during configuration. JTAG pin stability prevents accidental loading of JTAG instructions. Table 6–11 describes the dedicated JTAG pins.

Table 6–11. Dedicated JTAG Pins

Pin Name	User Mode	Pin Type	Description
TDI	N/A	Input	Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}.
TDO	N/A	Output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by leaving this pin unconnected.
TMS	N/A	Input	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}.
TCK	N/A	Input	The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND.
TRST	N/A	Input	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to IEEE Std. 1149.1. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND.

7. Configuring APEX 20KE & APEX 20KC Devices

CF51005-2.2

Introduction

APEX™ 20KE and APEX 20KC devices can be configured using one of four configuration schemes. All configuration schemes use either a microprocessor or configuration device.

This section covers how to configure APEX 20KE and APEX 20KC Devices, which use a 1.8-V voltage supply for V_{CCINT}. APEX 20K (non-E and non-C) devices use a 2.5-V voltage supply for the core. If your target FPGA is an APEX 20K device which uses a 2.5-V V_{CCINT}, refer to *Configuring Mercury, APEX 20K (2.5 V), ACEX 1K and FLEX 10K Devices* in the *Configuration Handbook*.

APEX 20KE and APEX 20KC devices can be configured using the passive serial (PS), passive parallel synchronous (PPS), passive parallel asynchronous (PPA), and Joint Test Action Group (JTAG) configuration schemes. The configuration scheme used is selected by driving the APEX 20KE or APEX 20KC device `MSEL1` and `MSEL0` pins either high or low as shown in Table 7–1. If your application only requires a single configuration mode, the `MSEL` pins can be connected to V_{CC} (V_{CCIO} of the I/O bank where the `MSEL` pin resides) or to ground. If your application requires more than one configuration mode, you can switch the `MSEL` pins after the FPGA is configured successfully. Toggling these pins during user-mode does not affect the device operation; however, the `MSEL` pins must be valid before a reconfiguration is initiated.

Table 7–1. APEX 20KE & APEX 20KC Configuration Schemes

MSEL1	MSEL0	Configuration Scheme
0	0	PS
1	0	PPS
1	1	PPA
(1)	*(1)*	JTAG Based *(2)*

Notes to Table 7–1:

(1) Do not leave the `MSEL` pins floating; connect them to a low- or high-logic level. These pins support the non-JTAG configuration scheme used in production. If only JTAG configuration is used, you should connect the `MSEL` pins to ground.

(2) JTAG-based configuration takes precedence over other configuration schemes, which means `MSEL` pin settings are ignored.

Table 7–2 shows the approximate configuration file sizes for APEX 20KE and APEX 20KE devices.

Table 7–2. APEX 20KE & APEX 20KC Raw Binary File (.rbf) Sizes

Device	Data Size (Bits)	Data Size (Bytes)
EP20K30E	354,832	44,354
EP20K60E	648,016	81,002
EP20K100E	1,008,016	126,002
EP20K160E	1,524,016	190,502
EP20K200E EP20K200C	1,968,016	246,002
EP20K300E	2,741,616	342,702
EP20K400E EP20K400C	3,909,776	488,722
EP20K600E EP20K600C	5,673,936	709,242
EP20K1000E EP20K1000C	8,960,016	1,120,002
EP20K1500E	12,042,256	1,505,282

Use the data in Table 7–2 only to estimate the file size before design compilation. Different configuration file formats, such as a Hexidecimal (**.hex**) or Tabular Text File (**.ttf**) format, will have different file sizes. However, for any specific version of the Quartus® II or MAX+PLUS® II software, all designs targeted for the same device will have the same configuration file size.

The following sections describe in detail how to configure APEX 20KE and APEX 20KC devices using the supported configuration schemes. The Device Configuration Pins section describes the device configuration pins available. The last section applies only to APEX 20KE devices and provides guidelines that you must follow to ensure successful configuration upon power-up and recovery from brown-out conditions. In this chapter, the generic term "device(s)" or "FPGA(s)" will include all APEX 20KE and APEX 20KC devices.

For more information on setting device configuration options or creating configuration files, refer to *Software Settings*, chapter 6 and 7 in volume 2 of the *Configuration Handbook*.

Passive Serial Configuration

You can perform APEX 20KE and APEX 20KC PS configuration using an Altera configuration device, an intelligent host (e.g., a microprocessor or Altera® MAX® device), or a download cable.

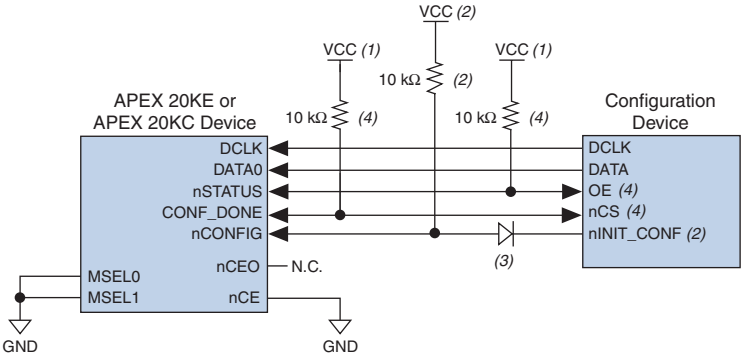
PS Configuration Using a Configuration Device

You can use an Altera configuration device, such as an enhanced configuration device, EPC2, or EPC1 device, to configure APEX 20KE and APEX 20KC devices using a serial configuration bitstream. Configuration data is stored in the configuration device. Figure 7–1 shows the configuration interface connections between the APEX 20KE or APEX 20KC device and a configuration device for single device configuration.

The figures in this chapter only show the configuration-related pins and the configuration pin connections between the configuration device and the FPGA.

For more information on the enhanced configuration device and flash interface pins (e.g., PGM[2..0], EXCLK, PORSEL, A[20..0], and DQ[15..0]), refer to the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* in the *Configuration Handbook*.

Figure 7–1. Single Device PS Configuration Using a Configuration Device



Notes to Figure 7–1:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CC} through a 10-kΩ resistor. For APEX 20KE devices, nCONFIG should be pulled up to V_{CCINT}. For APEX 20KC devices, nCONFIG should be connected to the same supply voltage as the configuration device.

(3) The nINIT_CONF pin has an internal pull-up resistor to 3.3 V that is always active. Since a 10-kΩ pull-up resistor to V_{CCINT} is required to successfully configure APEX 20KE devices, you need to isolate the 1.8-V V_{CCINT} from the configuration device's 3.3-V supply. To isolate the 1.8-V and 3.3-V power supplies, add a diode between the APEX 20KE device's nCONFIG pin and the configuration device's nINIT_CONF pin. Select a diode with a threshold voltage (V_T) less than or equal to 0.7 V. The diode will make the nINIT_CONF pin an open-drain pin; the pin will only be able to drive low or tri-state. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), this diode is not needed. The diode is also not needed when configuring APEX 20KC devices.

(4) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. For successful configuration of APEX 20KE and APEX 20KC devices using EPC2 devices, you should use external 10-kΩ pull-up resistors. If internal pull-up resistors on the enhanced configuration device are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the *Disable nCS and OE pull-up resistors on configuration device* option when generating programming files.

The value of the internal pull-up resistors on the enhanced configuration devices and EPC2 devices can be found in the Operating Conditions table of the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* or the *Configuration Devices for SRAM-based LUT Devices Data Sheet* in the *Configuration Handbook*.

When using enhanced configuration devices or EPC2 devices, nCONFIG of the FPGA can be connected to nINIT_CONF, which allows the INIT_CONF JTAG instruction to initiate FPGA configuration. The nINIT_CONF pin does not need to be connected if its functionality is not used. An internal pull-up resistor on the nINIT_CONF pin is always active in the enhanced configuration devices and the EPC2 devices, which means an external pull-up resistor is not required if nCONFIG is tied to nINIT_CONF. Since a 10-kΩ pull-up resistor to V_{CCINT} is required to successfully configure APEX 20KE devices, you need to isolate the 1.8-V V_{CCINT} from the configuration device's 3.3-V supply. To isolate the 1.8-V and 3.3-V power supplies, add a diode between the APEX 20KE device's nCONFIG pin and the configuration device's nINIT_CONF pin. Select a diode with a threshold voltage (V_T) less than or equal to 0.7 V. The diode will make the nINIT_CONF pin an open-drain pin; the pin will only be able to drive low or tri-state. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CCINT} through a 10-kΩ pull-up resistor and the isolating diode is not needed.

Upon power-up, the APEX 20KE or APEX 20KC device goes through a Power-On Reset (POR) for approximately 5 µs. During POR, the device resets and holds nSTATUS low, and tri-states all user I/O pins. The configuration device also goes through a POR delay to allow the power supply to stabilize. The POR time for EPC2, EPC1, and EPC1441 devices is 200 ms (maximum), and for enhanced configuration devices, the POR time can be set to either 100 ms or 2 ms, depending on its PORSEL pin setting. If the PORSEL pin is connected to GND, the POR delay is 100 ms. During this time, the configuration device drives its OE pin low. This low signal delays configuration because the OE pin is connected to the target device's nSTATUS pin. When both devices complete POR, they release their open-drain OE or nSTATUS pin, which is then pulled high by a pull-up resistor. Once the FPGA successfully exits POR, all user I/O pins are tri-stated. APEX 20KE and APEX 20KC devices have weak pull-up resistors on the user I/O pins which are on before and during configuration.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the Operating Conditions table of the *APEX 20K Programmable Logic Device Family Data Sheet* or *APEX 20KC Programmable Logic Device Family Data Sheet*.

When the power supplies have reached the appropriate operating voltages, the target FPGA senses the low-to-high transition on nCONFIG and initiates the configuration cycle. The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. The beginning of configuration can be delayed by holding the nCONFIG or nSTATUS pin low.

V_{CCINT} and V_{CCIO} of the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the nSTATUS pin, which is pulled high by a pull-up resistor. Enhanced configuration and EPC2 devices have an optional internal pull-up on the OE pin. This option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. For successful configuration of APEX 20KE and APEX 20KC devices using EPC2 devices, you should use external 10-kΩ pull-up resistors. If internal pull-up resistors on the enhanced configuration device are used, an external 10-kΩ pull-up resistor on the nCS/CONF_DONE line is not required. Once nSTATUS is released, the FPGA is ready to receive configuration data and the configuration stage begins.

When nSTATUS is pulled high, OE of the configuration device also goes high and the configuration device clocks data out serially to the FPGA using its internal oscillator. The APEX 20KE or APEX 20KC device receives configuration data on its DATA0 pin and the clock is received on the DCLK pin. Data is latched into the FPGA on the rising edge of DCLK.

After the FPGA has received all configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by a pull-up resistor. Since CONF_DONE is tied to the configuration device's nCS pin, the configuration device is disabled when CONF_DONE goes high. Enhanced configuration and EPC2 devices have an optional internal pull-up resistor on the nCS pin. This option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. For successful configuration of APEX 20KE and APEX 20KC devices using EPC2 devices, you should use external 10-kΩ pull-up resistors. If internal pull-up resistors on the enhanced configuration device are used, an external 10-kΩ pull-up resistor on the nCS/CONF_DONE line is not required. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin.

In APEX 20KE and APEX 20KC devices, the initialization clock source is either the APEX 20KE or APEX 20KC internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the APEX 20KE or APEX 20KC device will supply itself with enough clock cycles for proper initialization. You also have the flexibility to synchronize initialization of multiple devices by using the CLKUSR option. You can turn on the *Enable user-supplied start-up clock (CLKUSR)* option in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR will not affect the

configuration process. After all configuration data is accepted and CONF_DONE goes high, APEX 20KE and APEX 20KC devices require 40 clock cycles to properly initialize.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The *Enable INIT_DONE output* option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used, it will be high due to an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. This low-to-high transition signals that the FPGA has entered user mode. In user-mode, the user I/O pins will no longer have weak pull-up resistors and will function as assigned in your design. The enhanced configuration device and EPC2 device drives DCLK low and DATA high (EPC1 devices tri-state DATA) at the end of configuration.

If an error occurs during configuration, the FPGA drives its nSTATUS pin low, resetting itself internally. Since the nSTATUS pin is tied to OE, the configuration device will also be reset. If the *Auto-Restart Configuration After Error* option available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box is turned on, the FPGA automatically initiates reconfiguration if an error occurs. The APEX 20KE or APEX 20KC device will release its nSTATUS pin after a reset time-out period (maximum of 40 µs). When the nSTATUS pin is released and pulled high by a pull-up resistor, the configuration device reconfigures the chain. If this option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 8 µs to restart configuration. The external system can pulse nCONFIG if nCONFIG is under system control rather than tied to V_{CC}.

In addition, if the configuration device sends all of its data and then detects that CONF_DONE has not gone high, it recognizes that the FPGA has not configured successfully. Enhanced configuration devices wait for 64 DCLK cycles after the last configuration bit was sent for CONF_DONE to reach a high state. EPC1 and EPC2 devices wait for 16 DCLK cycles. In this case, the configuration device pulls its OE pin low, which in turn drives the target device's nSTATUS pin low. If the *Auto-Restart Configuration After Error* option is set in the software, the target device resets and then releases its nSTATUS pin after a reset time-out period (maximum of 40 µs). When nSTATUS returns high, the configuration device tries to reconfigure the FPGA.

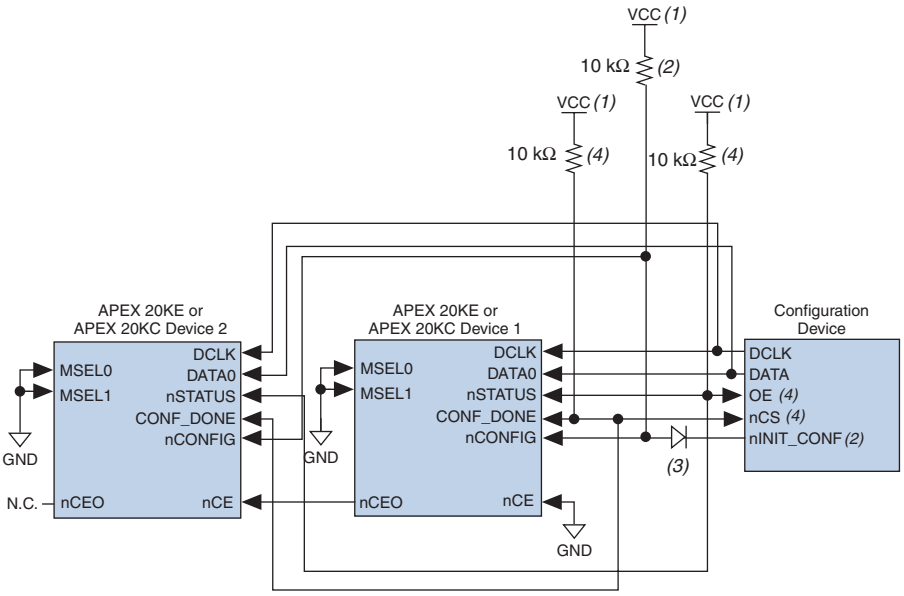
When CONF_DONE is sensed low after configuration, the configuration device recognizes that the target device has not configured successfully; therefore, your system should not pull CONF_DONE low to delay initialization. Instead, use the CLKUSR option to synchronize the initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain will initialize together if their CONF_DONE pins are tied together.

If the optional CLKUSR pin is being used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure that CLKUSR continues toggling during the time nSTATUS is low (maximum of 40 µs).

When the FPGA is in user-mode, a reconfiguration can be initiated by pulling the nCONFIG pin low. The nCONFIG pin should be low for at least 8 µs. When nCONFIG is pulled low, the FPGA also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Since CONF_DONE is pulled low, this will activate the configuration device since it will see its nCS pin drive low. Once nCONFIG returns to a logic high state and nSTATUS is released by the FPGA, reconfiguration begins.

Figure 7–2 shows how to configure multiple devices with a configuration device. This circuit is similar to the configuration device circuit for a single device, except the APEX 20KE or APEX 20KC devices are cascaded for multi-device configuration.

Figure 7–2. Multi-Device PS Configuration Using a Configuration Device



Notes to Figure 7–2:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CC} through a 10-kΩ resistor. For APEX 20KE devices, nCONFIG should be pulled up to V_{CCINT}. For APEX 20KC devices, nCONFIG should be connected to the same supply voltage as the configuration device.

(3) The nINIT_CONF pin has an internal pull-up resistor to 3.3 V that is always active. Since a 10-kΩ pull-up to V_{CCINT} is required to successfully configure APEX 20KE devices, you need to isolate the 1.8-V V_{CCINT} from the configuration device's 3.3-V supply. To isolate the 1.8-V and 3.3-V power supplies, add a diode between the APEX 20KE device's nCONFIG pin and the configuration device's nINIT_CONF pin. Select a diode with a threshold voltage (V_T) less than or equal to 0.7 V. The diode will make the nINIT_CONF pin an open-drain pin; the pin will only be able to drive low or tri-state. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), this diode is not needed. The diode is also not needed when configuring APEX 20KC devices.

(4) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. For successful configuration of APEX 20KE and APEX 20KC devices using EPC2 devices, you should use external 10-kΩ pull-up resistors. If internal pull-up resistors on the enhanced configuration device are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the *Disable nCS and OE pull-up resistors on configuration device* option when generating programming files.

When performing multi-device configuration, you must generate the configuration device's Programmer Object File (**.pof**) from each project's SRAM Object File (**.sof**). You can combine multiple SOFs using the Quartus II software.

For more information on how to create configuration files for multi-device configuration chains, refer to *Software Settings*, chapter 6 and 7 in volume 2 of the *Configuration Handbook*.

In multi-device PS configuration, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they can require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DCLK and DATA lines are buffered for every fourth device.

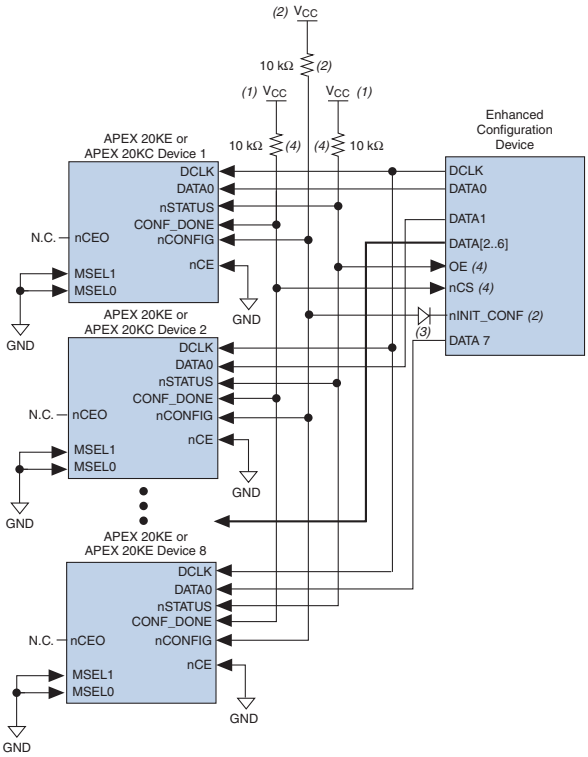
When configuring multiple devices, configuration does not begin until all devices release their OE or nSTATUS pins. Similarly, since all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Since all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first FPGA flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This low signal drives the OE pin low on the configuration device and drives nSTATUS low on all FPGAs, which causes them to enter a reset state. This behavior is similar to a single FPGA detecting an error.

If the *Auto-Restart Configuration After Error* option is turned on, the devices will automatically initiate reconfiguration if an error occurs. The FPGAs will release their nSTATUS pins after a reset time-out period (maximum of 40 µs). When all the nSTATUS pins are released and pulled high, the configuration device tries to reconfigure the chain. If the *Auto-Restart Configuration After Error* option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 8 µs to restart configuration. The external system can pulse nCONFIG if nCONFIG is under system control rather than tied to V_{CC}.

The enhanced configuration devices also support parallel configuration of up to eight devices. The n-bit (n = 1, 2, 4, or 8) PS configuration mode allows enhanced configuration devices to concurrently configure FPGAs or a chain of FPGAs. In addition, these devices do not have to be the same device family or density; they can be any combination of Altera FPGAs. An individual enhanced configuration device DATA line is available for each targeted FPGA. Each DATA line can also feed a daisy chain of FPGAs. Figure 7–3 shows how to concurrently configure multiple devices using an enhanced configuration device.

Figure 7–3. Concurrent PS Configuration of Multiple Devices Using an Enhanced Configuration Device



Notes to Figure 7–3:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CC} through a 10-kΩ resistor. For APEX 20KE devices, nCONFIG should be pulled up to V_{CCINT}. For APEX 20KC devices, nCONFIG should be connected to the same supply voltage as the configuration device.

(3) The nINIT_CONF pin has an internal pull-up resistor to 3.3 V that is always active. Since a 10-kΩ pull-up to V_{CCINT} is required to successfully configure APEX 20KE devices, you need to isolate the 1.8-V V_{CCINT} from the configuration device's 3.3-V supply. To isolate the 1.8-V and 3.3-V power supplies, add a diode between the APEX 20KE device's nCONFIG pin and the configuration device's nINIT_CONF pin. Select a diode with a threshold voltage (V_T) less than or equal to 0.7 V. The diode will make the nINIT_CONF pin an open-drain pin; the pin will only be able to drive low or tri-state. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), this diode is not needed. The diode is also not needed when configuring APEX 20KC devices.

(4) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. For successful configuration of APEX 20KE and APEX 20KC devices using EPC2 devices, you should use external 10-kΩ pull-up resistors. If internal pull-up resistors on the enhanced configuration device are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the *Disable nCS and OE pull-up resistors on configuration device* option when generating programming files.

The Quartus II software only allows the selection of n-bit PS configuration modes, where n must be 1, 2, 4, or 8. However, you can use these modes to configure any number of devices from 1 to 8. When configuring SRAM-based devices using n-bit PS modes, use Table 7–3 to select the appropriate configuration mode for the fastest configuration times.

Table 7–3. Recommended Configuration Using n-Bit PS Modes

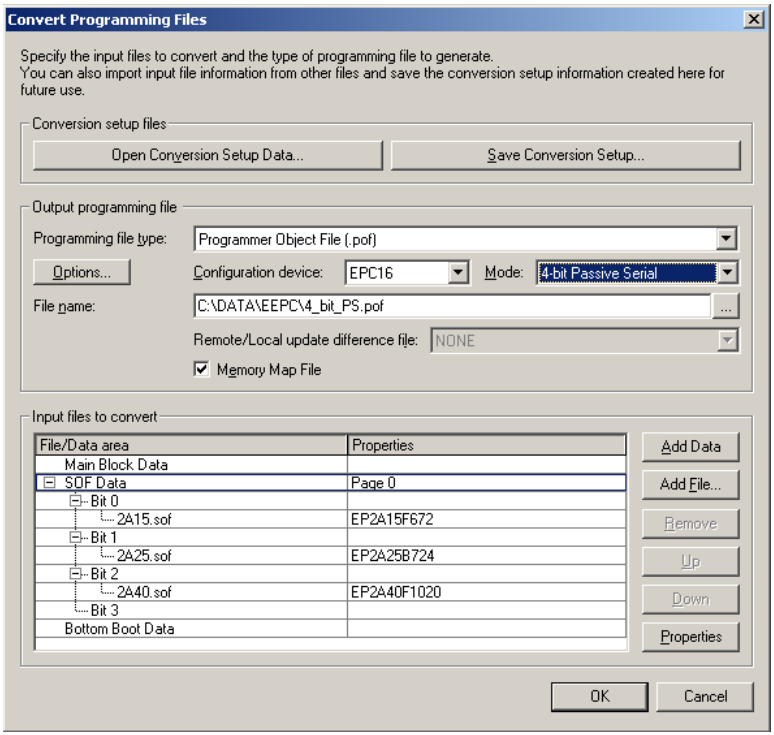
Number of Devices *(1)*	Recommended Configuration Mode
1	1-bit PS
2	2-bit PS
3	4-bit PS
4	4-bit PS
5	8-bit PS
6	8-bit PS
7	8-bit PS
8	8-bit PS

Note to Table 7–3:

(1) Assume that each DATA line is only configuring one device, not a daisy chain of devices.

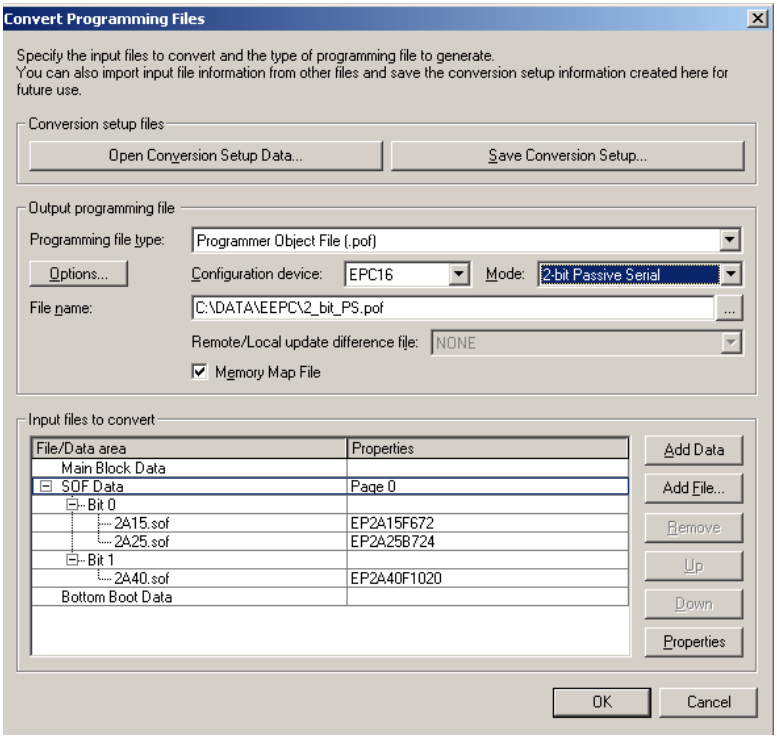
For example, if you configure three FPGAs, you would use the 4-bit PS mode. For the DATA0, DATA1, and DATA2 lines, the corresponding SOF data is transmitted from the configuration device to the FPGA. For DATA3, you can leave the corresponding Bit3 line blank in the Quartus II software. On the printed circuit board (PCB), leave the DATA3 line from the enhanced configuration device unconnected. Figure 7–4 shows the Quartus II **Convert Programming Files** window (Tools menu) setup for this scheme.

Figure 7–4. Software Settings for Configuring Devices Using n-Bit PS Modes



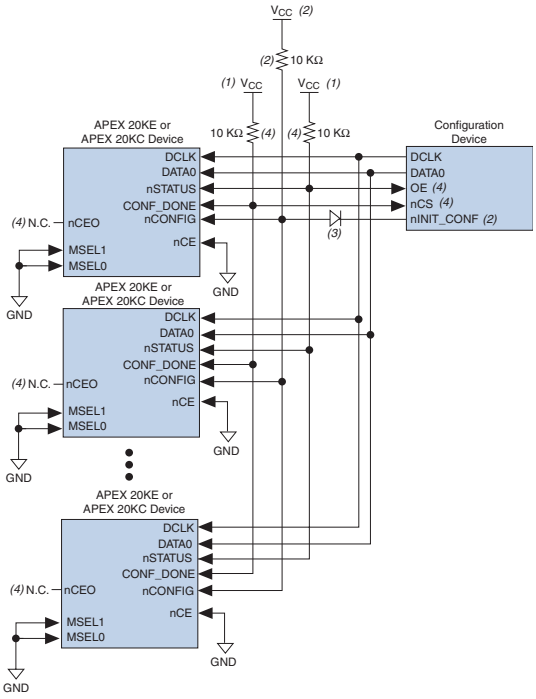
Alternatively, you can daisy chain two FPGAs to one DATA line while the other DATA lines drive one device each. For example, you could use the 2-bit PS mode to drive two FPGAs with DATA Bit0 (EP20K400E and EP20K600E devices) and the third device (the EP20K1000E device) with DATA Bit1. This 2-bit PS configuration scheme requires less space in the configuration flash memory, but can increase the total system configuration time (Figure 7–5).

Figure 7–5. Software Settings for Daisy Chaining Two FPGAs on One DATA Line



In your system, you may have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they can require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete configuration at the same time. Figure 7–6 shows multi-device PS configuration when the APEX 20KE and APEX 20KC devices are receiving the same configuration data.

Figure 7–6. Multiple-Device PS Configuration Using an Enhanced Configuration Device When FPGAs Receive the Same Data



Notes to Figure 7–6:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CC} through a 10-kΩ resistor. For APEX 20KE devices, nCONFIG should be pulled up to V_{CCINT}. For APEX 20KC devices, nCONFIG should be connected to the same supply voltage as the configuration device.

(3) The nINIT_CONF pin has an internal pull-up resistor to 3.3 V that is always active. Since a 10-kΩ pull-up to V_{CCINT} is required to successfully configure APEX 20KE devices, you need to isolate the 1.8-V V_{CCINT} from the configuration device's 3.3-V supply. To isolate the 1.8-V and 3.3-V power supplies, add a diode between the APEX 20KE device's nCONFIG pin and the configuration device's nINIT_CONF pin. Select a diode with a threshold voltage (V_T) less than or equal to 0.7 V. The diode will make the nINIT_CONF pin an open-drain pin; the pin will only be able to drive low or tri-state. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), this diode is not needed. The diode is also not needed when configuring APEX 20KC devices.

(4) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. For successful configuration of APEX 20KE and APEX 20KC devices using EPC2 devices, you should use external 10-kΩ pull-up resistors. If internal pull-up resistors on the enhanced configuration device are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the *Disable nCS and OE pull-up resistors on configuration device* option when generating programming files.

(5) The nCEO pins of all devices are left unconnected when configuring the same configuration data into multiple devices.

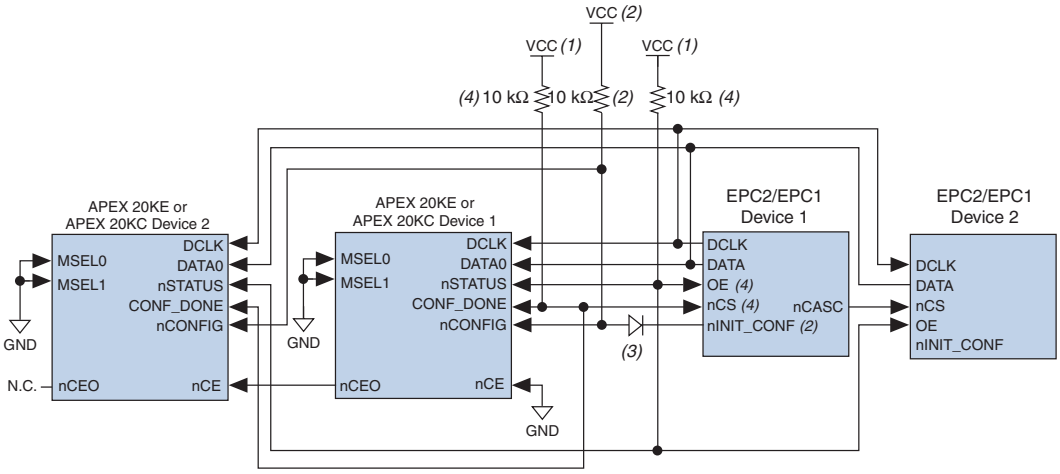
You can cascade several EPC2 or EPC1 devices to configure multiple APEX 20KE or APEX 20KC devices. The first configuration device in the chain is the master configuration device, while the subsequent devices are the slave devices. The master configuration device sends DCLK to the APEX 20KE and APEX 20KC devices and to the slave configuration devices. The first EPC device's nCS pin is connected to the CONF_DONE pins of the FPGAs, while its nCASC pin is connected to nCS of the next configuration device in the chain. The last device's nCS input comes from the previous device, while its nCASC pin is left floating. When all data from the first configuration device is sent, it drives nCASC low, which in turn drives nCS on the next configuration device. Because a configuration device requires less than one clock cycle to activate a subsequent configuration device, the data stream is uninterrupted.

Enhanced configuration devices EPC4, EPC8, and EPC16 cannot be cascaded.

Since all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, the master configuration device stops configuration for the entire chain and the entire chain must be reconfigured. For example, if the master configuration device does not detect CONF_DONE going high at the end of configuration, it resets the entire chain by pulling its OE pin low. This low signal drives the OE pin low on the slave configuration device(s) and drives nSTATUS low on all FPGAs, causing them to enter a reset state. This behavior is similar to the FPGA detecting an error in the configuration data.

Figure 7–7 shows how to configure multiple devices using cascaded EPC2 or EPC1 devices.

Figure 7–7. Multi-Device PS Configuration Using Cascaded EPC2 or EPC1 Devices



Notes to Figure 7–7:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CC} through a 10-kΩresistor. For APEX 20KE devices, nCONFIG should be pulled up to V_{CCINT}. For APEX 20KC devices, nCONFIG should be connected to the same supply voltage as the configuration device.

(3) The nINIT_CONF pin has an internal pull-up resistor to 3.3 V that is always active. Since a 10-kΩpull-up resistor to V_{CCINT} is required to successfully configure APEX 20KE devices, you need to isolate the 1.8-V V_{CCINT} from the configuration device's 3.3-V supply. To isolate the 1.8-V and 3.3-V power supplies, add a diode between the APEX 20KE device's nCONFIG pin and the configuration device's nINIT_CONF pin. Select a diode with a threshold voltage (V_T) less than or equal to 0.7 V. The diode will make the nINIT_CONF pin an open-drain pin; the pin will only be able to drive low or tri-state. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), this diode is not needed. The diode is also not needed when configuring APEX 20KC devices.

(4) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. For successful configuration of APEX 20KE and APEX 20KC devices using EPC2 devices, you should use external 10-kΩpull-up resistors. If internal pull-up resistors on the enhanced configuration device are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the *Disable nCS and OE pull-up resistors on configuration device* option when generating programming files.

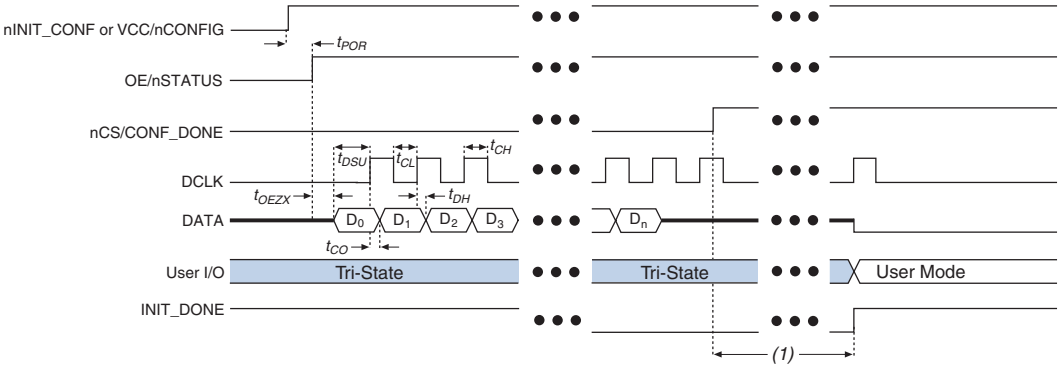
When using enhanced configuration devices or EPC2 devices, nCONFIG of the FPGA can be connected to nINIT_CONF, which allows the INIT_CONF JTAG instruction to initiate FPGA configuration. The nINIT_CONF pin does not need to be connected if its functionality is not used. An internal pull-up resistor on the nINIT_CONF pin is always active in the enhanced configuration devices and the EPC2 devices, which means an external pull-up resistor is not required if nCONFIG is tied to nINIT_CONF. Since a 10-kΩ pull-up resistor to V_{CCINT} is required to successfully configure APEX 20KE devices, you need to isolate the 1.8-V V_{CCINT} from the configuration device's 3.3-V supply. To isolate the 1.8-V and 3.3-V power supplies, add a diode between the APEX 20KE device's nCONFIG pin and the configuration device's nINIT_CONF pin. Select a diode with a threshold voltage (V_T) less than or equal to 0.7 V. The diode will make the nINIT_CONF pin an open-drain pin; the pin will only be able to drive low or tri-state. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CCINT} through a 10 kΩ resistor and the isolating diode is not needed. If multiple EPC2 devices are used to configure an APEX 20KE or APEX 20KC device(s), only the first EPC2 has its nINIT_CONF pin tied to the device's nCONFIG pin.

You can use a single configuration chain to configure APEX 20KE and APEX 20KC devices with other Altera devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information on configuring multiple Altera devices in the same configuration chain, refer to *Configuring Mixed Altera FPGA Chains* in the *Configuration Handbook*.

Figure 7–8 shows the timing waveform for the PS configuration scheme using a configuration device.

Figure 7–8. APEX 20KE & APEX 20KC PS Configuration Using a Configuration Device Timing Waveform



Note to Figure 7–8:

(1) APEX 20KE and APEX 20KC devices enter user-mode 40 clock cycles after CONF_DONE goes high. The initialization clock can come from the APEX 20KE or APEX 20KC internal oscillator or the CLKUSR pin.

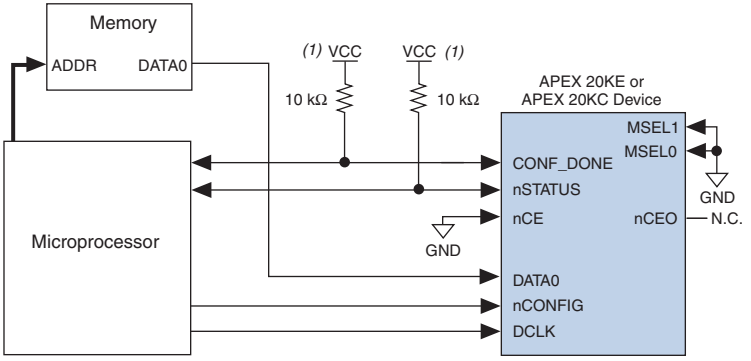
For timing information, refer to the *Enhanced Configuration Devices (EPC4, EPC8, and EPC16) Data Sheet* or the *Configuration Devices for SRAM-based LUT Devices Data Sheet* in the Configuration Handbook.

Device configuration options and how to create configuration files are discussed further in *Software Settings*, ***chapter 6 and 7 in volume 2*** **of** the *Configuration Handbook*.

PS Configuration Using a Microprocessor

In the PS configuration scheme, an intelligent host (e.g., a microprocessor or CPLD) can transfer configuration data from a storage device (e.g., flash memory) to the target APEX 20KE and APEX 20KC devices. Configuration data can be stored in RBF, HEX, or TTF format. Figure 7–9 shows the configuration interface connections between the APEX 20KE or APEX 20KC device and a microprocessor for single device configuration.

Figure 7–9. Single Device PS Configuration Using a Microprocessor



Note to Figure 7–9:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for the device.

Upon power-up, the APEX 20KE or APEX 20KC device goes through a POR for approximately 5 µs. During POR, the device resets and holds nSTATUS low, and tri-states all user I/O pins. Once the FPGA successfully exits POR, all user I/O pins are tri-stated. APEX 20KE and APEX 20KC devices have weak pull-up resistors on the user I/O pins which are on before and during configuration.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the Operating Conditions table of the *APEX 20K Programmable Logic Device Family Data Sheet* or *APEX 20KC Programmable Logic Device Family Data Sheet*.

The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration, the microprocessor must generate a low-to-high transition on the nCONFIG pin.

VCCINT and VCCIO pins on the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. Once nSTATUS is released, the FPGA is ready to receive configuration data and the configuration stage begins. When nSTATUS is

pulled high, the microprocessor should place the configuration data one bit at a time on the DATA0 pin. The least significant bit (LSB) of each data byte must be sent first.

The APEX 20KE or APEX 20KC device receives configuration data on its DATA0 pin and the clock is received on the DCLK pin. Data is latched into the FPGA on the rising edge of DCLK. Data is continuously clocked into the target device until CONF_DONE goes high. After the FPGA has received all configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by an external 10-kΩ pull-up resistor. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin.

In APEX 20KE and APEX 20KC devices, the initialization clock source is either the APEX 20KE or APEX 20KC internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the APEX 20KE or APEX 20KC device will take care to provide itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device. Driving DCLK to the device after configuration is complete does not affect device operation.

You also have the flexibility to synchronize initialization of multiple devices by using the CLKUSR option. The *Enable user-supplied start-up clock (CLKUSR)* option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR will not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, APEX 20KE and APEX 20KC devices require 40 clock cycles to initialize properly.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The *Enable INIT_DONE output* option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used it will be high due to an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low. When initialization is complete, the INIT_DONE pin will be released and pulled high. The microprocessor must be able to detect this low-to-high transition which signals the FPGA has entered user mode. In user-mode, the user I/O pins will no longer have weak pull-up resistors and will function as assigned in your design.

To ensure DCLK and DATA are not left floating at the end of configuration, the microprocessor must drive them either high or low, whichever is convenient on your board.

Handshaking signals are not used in PS configuration mode. Therefore, the configuration clock (DCLK) speed must be below the specified frequency to ensure correct configuration. No maximum DCLK period exists, which means you can pause configuration by halting DCLK for an indefinite amount of time.

If an error occurs during configuration, the FPGA drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the microprocessor that there is an error. If the *Auto-Restart Configuration After Error* option (available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box) is turned on, the APEX 20KE or APEX 20KC device releases nSTATUS after a reset time-out period (maximum of 40 µs). After nSTATUS is released and pulled high by a pull-up resistor, the microprocessor can try to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition (with a low pulse of at least 8 µs) on nCONFIG to restart the configuration process.

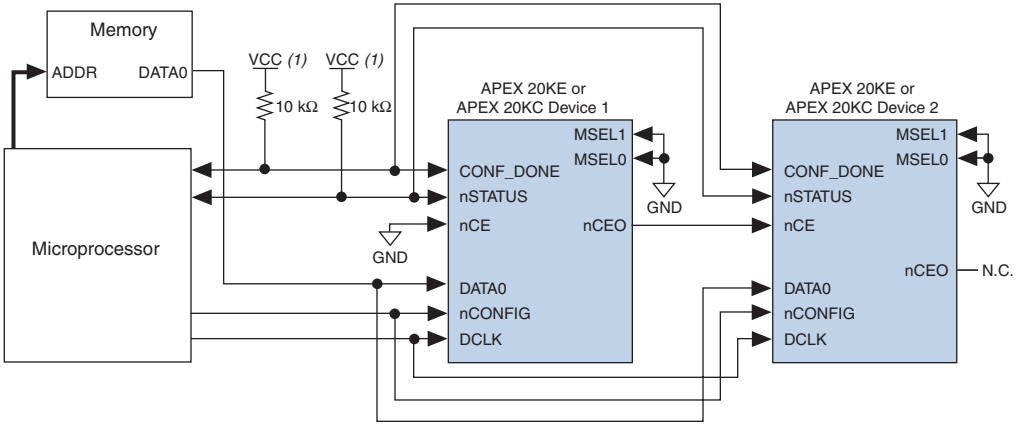
The microprocessor can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The CONF_DONE pin must be monitored by the microprocessor to detect errors and determine when programming completes. If the microprocessor sends all configuration data but CONF_DONE or INIT_DONE have not gone high, the microprocessor must reconfigure the target device.

If the optional CLKUSR pin is being used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure that CLKUSR continues toggling during the time nSTATUS is low (maximum of 40 µs).

When the FPGA is in user-mode, you can initiate a reconfiguration by transitioning the nCONFIG pin low-to-high. The nCONFIG pin must be low for at least 8 µs. When nCONFIG is pulled low, the FPGA also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Once nCONFIG returns to a logic high state and nSTATUS is released by the FPGA, reconfiguration begins.

Figure 7–10 shows how to configure multiple devices using a microprocessor. This circuit is similar to the PS configuration circuit for a single device, except the APEX 20KE or APEX 20KC devices are cascaded for multi-device configuration.

Figure 7–10. Multi-Device PS Configuration Using a Microprocessor



Note to Figure 7–10:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.

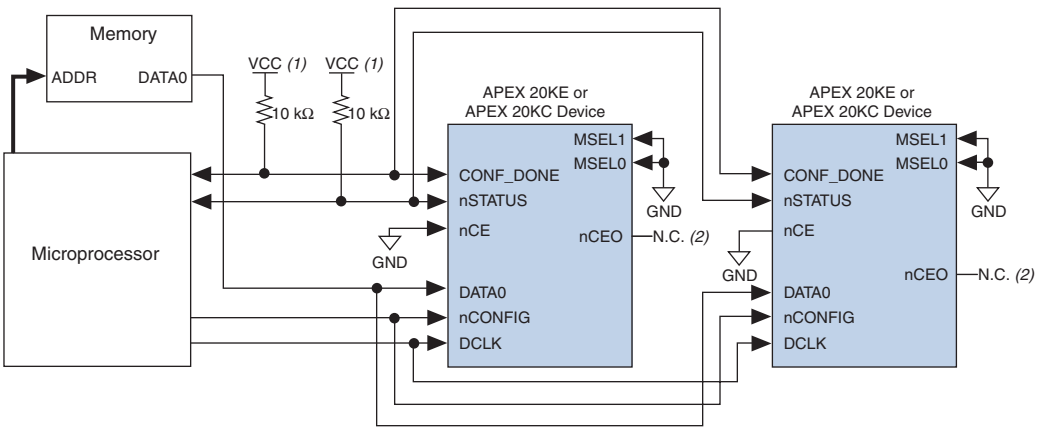
In multi-device PS configuration the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration within one clock cycle; therefore, the transfer of data destinations is transparent to the microprocessor. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they can require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DCLK and DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Since all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first FPGA flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single FPGA detecting an error.

If the *Auto-Restart Configuration After Error* option is turned on, the FPGAs release their nSTATUS pins after a reset time-out period (maximum of 40 µs). After all nSTATUS pins are released and pulled high, the microprocessor can try to reconfigure the chain without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition (with a low pulse of at least 8 µs) on nCONFIG to restart the configuration process.

In your system, you can have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they can require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete configuration at the same time. Figure 7–11 shows multi-device PS configuration when both APEX 20KE and APEX 20KC devices are receiving the same configuration data.

Figure 7–11. Multiple-Device PS Configuration Using a Microprocessor When Both FPGAs Receive the Same Data



Notes to Figure 7–11:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.

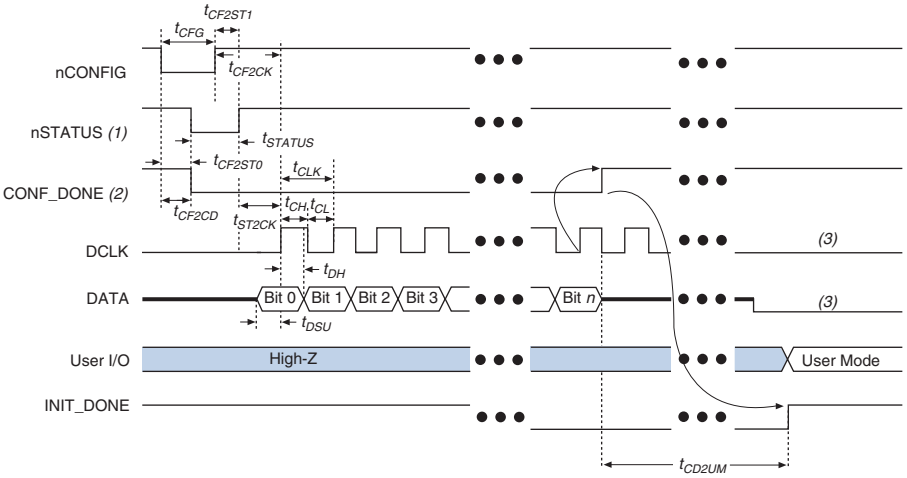
(2) The nCEO pins of both devices are left unconnected when configuring the same configuration data into multiple devices.

You can use a single configuration chain to configure APEX 20KE and APEX 20KC devices with other Altera devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information on configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* in the *Configuration Handbook*.

Figure 7–12 shows the timing waveform for the PS configuration for APEX 20KE and APEX 20KC devices using a microprocessor.

Figure 7–12. APEX 20KE & APEX 20KC PS Configuration Using a Microprocessor Timing Waveform



Notes to Figure 7–12:

(1) Upon power-up, the APEX 20KE or APEX 20KC device holds nSTATUS low for not more than 5 µs after V_{CC} reaches its minimum requirement.

(2) Upon power-up, before and during configuration, CONF_DONE is low.

(3) DATA0 and DCLK should not be left floating after configuration. It should be driven high or low, whichever is more convenient.

Table 7–4 defines the timing parameters for APEX 20KE and APEX 20KC devices for PS configuration.

Table 7–4. PS Timing Parameters for APEX 20KE & APEX 20KC Devices

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		200	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		200	ns
t_{CFG}	nCONFIG low pulse width	8		µs
t_{STATUS}	nSTATUS low pulse width	10	40 *(1)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high		1 *(1)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	40		µs
t_{ST2CK}	nSTATUS high to first rising edge on DCLK	1		µs
t_{DSU}	Data setup time before rising edge on DCLK	10		ns
t_{DH}	Data hold time after rising edge on DCLK	0		ns
t_{CH}	DCLK high time	7.5		ns
t_{CL}	DCLK low time	7.5		ns
t_{CLK}	DCLK period	15		ns
f_{MAX}	DCLK maximum frequency		66	MHz
t_{CD2UM}	CONF_DONE high to user mode *(2)*	2	8	µs

Notes to Table 7–4:

(1) This value is applicable if users do not delay configuration by extending the nSTATUS low pulse width.

(2) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting the device. If the clock source is CLKUSR, multiply the clock period by 40 for APEX 20KE and APEX 20KC devices to obtain this value.

Device configuration options and how to create configuration files are discussed further in *Software Settings*, chapter 6 and 7 in volume 2 of the *Configuration Handbook*.

Configuring Using the MicroBlaster Driver

The MicroBlaster™ software driver allows you to configure Altera's FPGAs through the ByteBlasterMV cable in PS mode. The MicroBlaster software driver supports a RBF programming input file and is targeted for embedded passive serial configuration. The source code is developed for the Windows NT operating system, although you can customize it to run on other operating systems. For more information on the MicroBlaster software driver, go to the Altera web site (**http://www.altera.com**).

PS Configuration Using a Download Cable

In this section, the generic term "download cable" includes the Altera USB Blaster universal serial bus (USB) port download cable, MasterBlaster™ serial/USB communications cable, ByteBlaster™ II parallel port download cable, and the ByteBlasterMV™ parallel port download cable.

In PS configuration with a download cable, an intelligent host (e.g., a PC) transfers data from a storage device to the FPGA via the USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV cable.

Upon power-up, the APEX 20KE or APEX 20KC device goes through a POR for approximately 5 µs. During POR, the device resets and holds nSTATUS low, and tri-states all user I/O pins. Once the FPGA successfully exits POR, all user I/O pins are tri-stated. APEX 20KE and APEX 20KC devices have weak pull-up resistors on the user I/O pins which are on before and during configuration.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the Operating Conditions table of the *APEX 20K Programmable Logic Device Family Data Sheet* or *APEX 20KC Programmable Logic Device Family Data Sheet*.

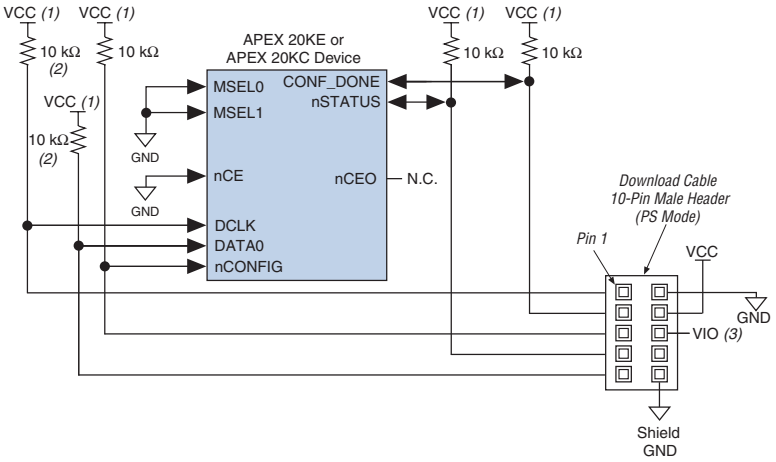
The configuration cycle consists of 3 stages: reset, configuration and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration in this scheme, the download cable generates a low-to-high transition on the nCONFIG pin.

VCCINT and VCCIO pins on the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. Once nSTATUS is released the FPGA is ready to receive configuration data and the configuration stage begins. The programming hardware or download cable then places the configuration data one bit at a time on the device's DATA0 pin. The configuration data is clocked into the target device until CONF_DONE goes high.

When using a download cable, setting the *Auto-Restart Configuration After Error* option does not affect the configuration cycle because you must manually restart configuration in the Quartus II software when an error occurs. Additionally, the *Enable user-supplied start-up clock (CLKUSR)* option has no affect on the device initialization since this option is disabled in the SOF when programming the FPGA using the Quartus II programmer and download cable. Therefore, if you turn on the CLKUSR option, you do not need to provide a clock on CLKUSR when you are configuring the FPGA with the Quartus II programmer and a download cable. Figure 7–13 shows PS configuration for APEX 20KE and APEX 20KC devices using a USB Blaster, MasterBlaster, ByteBlaster II or ByteBlasterMV cable.

Figure 7–13. PS Configuration Using a USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV Cable



Notes to Figure 7–13:

(1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster (VIO pin), ByteBlaster II or ByteBlasterMV cable.

(2) The pull-up resistors on DATA0 and DCLK are only needed if the download cable is the only configuration scheme used on your board. This is to ensure that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, the pull-up resistors on DATA0 and DCLK are not needed.

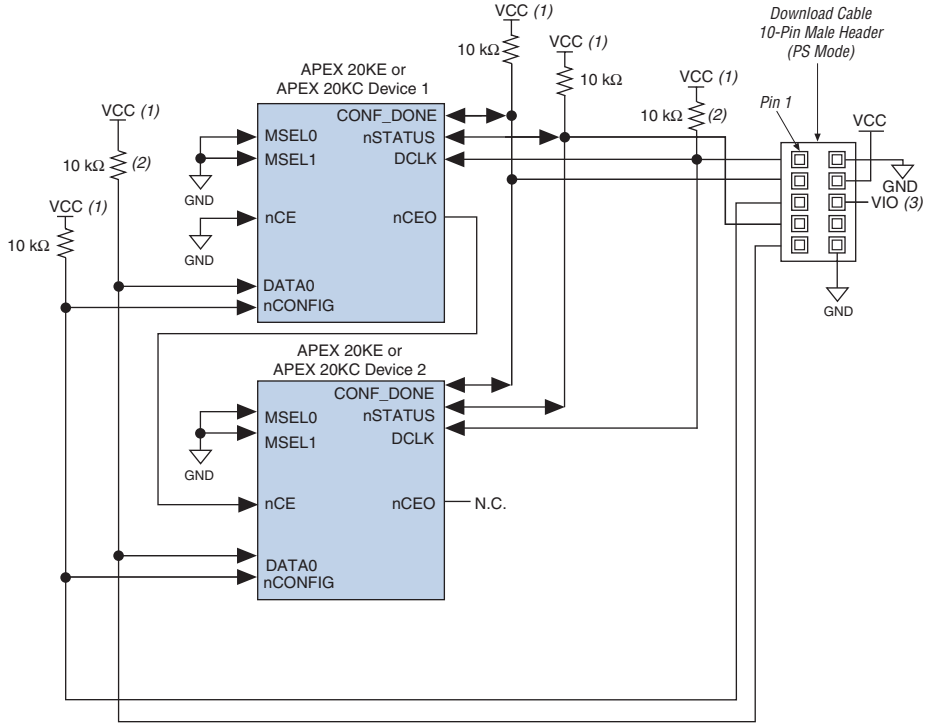
(3) Pin 6 of the header is a VIO reference voltage for the MasterBlaster output driver. VIO should match the device's V_{CCIO}. Refer to the MasterBlaster Serial/USB Communications Cable Data Sheet for this value. In the ByteBlasterMV, this pin is a no connect. In the USB Blaster and ByteBlaster II, this pin is connected to nCE when it is used for Active Serial programming, otherwise it is a no connect.

You can use a download cable to configure multiple APEX 20KE and APEX 20KC devices by connecting each device's nCEO pin to the subsequent device's nCE pin. The first device's nCE pin is connected to GND while its nCEO pin is connected to the nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. All other configuration pins, nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE are connected to every device in the chain. Because all CONF_DONE pins are tied together, all devices in the chain initialize and enter user mode at the same time.

In addition, because the nSTATUS pins are tied together, the entire chain halts configuration if any device detects an error. The *Auto-Restart Configuration After Error* option does not affect the configuration cycle because you must manually restart configuration in the Quartus II software when an error occurs.

Figure 7–14 shows how to configure multiple APEX 20KE and APEX 20KC devices with a download cable.

Figure 7–14. Multi-Device PS Configuration using a USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV Cable



Notes to Figure 7–14:

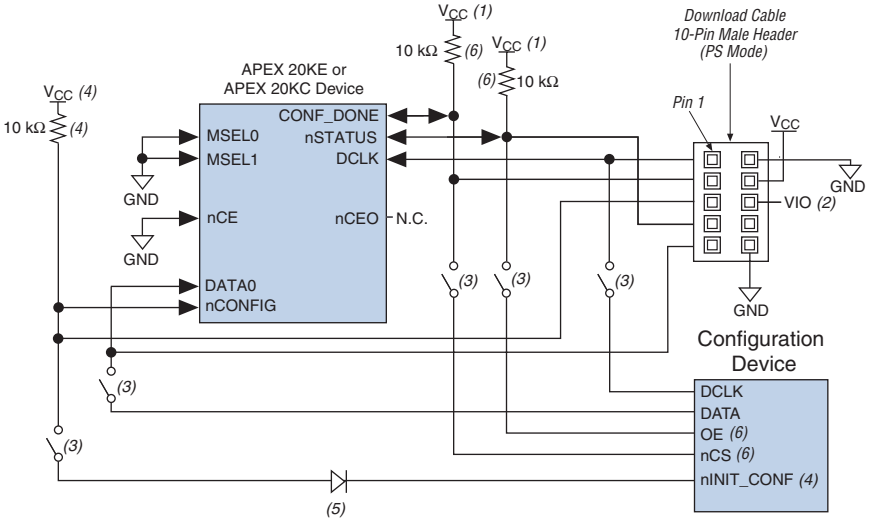
(1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster (VIO pin), ByteBlaster II, or ByteBlasterMV cable.

(2) The pull-up resistors on DATA0 and DCLK are only needed if the download cable is the only configuration scheme used on your board. This is to ensure that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, the pull-up resistors on DATA0 and DCLK are not needed.

(3) Pin 6 of the header is a VIO reference voltage for the MasterBlaster output driver. VIO should match the device's V_{CCIO}. Refer to the MasterBlaster Serial/USB Communications Cable Data Sheet for this value. In the ByteBlasterMV, this pin is a no connect. In the USB Blaster and ByteBlaster II, this pin is connected to nCE when it is used for Active Serial programming, otherwise it is a no connect.

If you are using a download cable to configure device(s) on a board that also has configuration devices, you should electrically isolate the configuration device from the target device(s) and cable. One way to isolate the configuration device is to add logic, such as a multiplexer, that can select between the configuration device and the cable. The multiplexer chip should allow bidirectional transfers on the nSTATUS and CONF_DONE signals. Another option is to add switches to the five common signals (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) between the cable and the configuration device. The last option is to remove the configuration device from the board when configuring the FPGA with the cable. Figure 7–15 shows a combination of a configuration device and a download cable to configure an FPGA.

Figure 7–15. PS Configuration with a Download Cable & Configuration Device Circuit



Notes to Figure 7–15:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) Pin 6 of the header is a VIO reference voltage for the MasterBlaster output driver. VIO should match the device's V_{CCIO}. Refer to the MasterBlaster Serial/USB Communications Cable Data Sheet for this value. In the ByteBlasterMV, this pin is a no connect. In the USB Blaster and ByteBlaster II, this pin is connected to nCE when it is used for Active Serial programming, otherwise it is a no connect.

(3) You should not attempt configuration with a download cable while a configuration device is connected to an APEX 20KE or APEX 20KC device. Instead, you should either remove the configuration device from its socket when using the download cable or place a switch on the five common signals between the download cable and the configuration device.

(4) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CC} through a 10-kΩ pull-up resistor. For APEX 20KE devices, nCONFIG should be pulled up to V_{CCINT}. For APEX 20KC devices, nCONFIG should be connected to the same supply voltage as the configuration device.

(5) The nINIT_CONF pin has an internal pull-up resistor to 3.3 V that is always active. Since a 10-kΩ pull-up to V_{CCINT} is required to successfully configure APEX 20KE devices, you need to isolate the 1.8-V V_{CCINT} from the configuration device's 3.3-V supply. To isolate the 1.8-V and 3.3-V power supplies, add a diode between the APEX 20KE device's nCONFIG pin and the configuration device's nINIT_CONF pin. Select a diode with a threshold voltage (V_T) less than or equal to 0.7 V. The diode will make the nINIT_CONF pin an open-drain pin; the pin will only be able to drive low or tri-state. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), this diode is not needed. The diode is also not needed when configuring APEX 20KC devices.

(6) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. For successful configuration of APEX 20KE and APEX 20KC devices using EPC2 devices, you should use external 10-kΩ pull-up resistors. If internal pull-up resistors on the enhanced configuration device are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the *Disable nCS and OE pull-up resistors on configuration device* option when generating programming files.

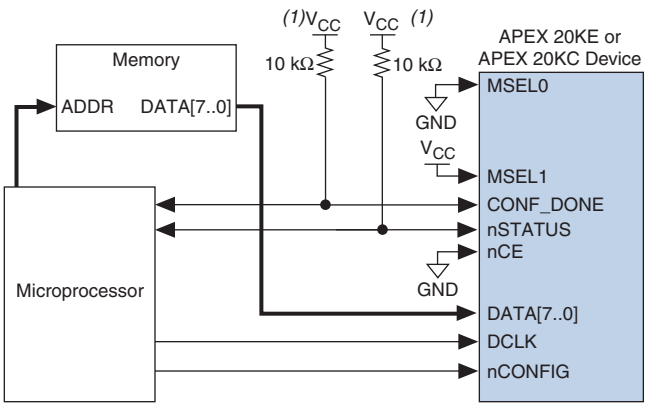
For more information on how to use the USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV cables, refer to the following data sheets:

- USB Blaster USB Port Download Cable Data Sheet
- MasterBlaster Serial/USB Communications Cable Data Sheet
- ByteBlaster II Parallel Port Download Cable Data Sheet
- ByteBlasterMV Parallel Port Download Cable Data Sheet

Passive Parallel Synchronous Configuration

Passive Parallel Synchronous (PPS) configuration uses an intelligent host, such as a microprocessor, to transfer configuration data from a storage device, such as flash memory, to the target APEX 20KE or APEX 20KC device. Configuration data can be stored in TTF, RBF, or HEX format. The host system outputs byte-wide data and the serializing clock to the FPGA. The target device latches the byte-wide data on the DATA[7..0] pins on the rising edge of DCLK and then uses the next eight falling edges on DCLK to serialize the data internally. On the ninth rising DCLK edge, the next byte of configuration data is latched into the target device. Figure 7–16 shows the configuration interface connections between the FPGA and a microprocessor for single device configuration.

Figure 7–16. Single Device PPS Configuration Using a Microprocessor



Note to Figure 7–16:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for the device.

Upon power-up, the APEX 20KE or APEX 20KC device goes through a Power-On Reset (POR) for approximately 5 µs. During POR, the device resets and holds nSTATUS low, and tri-states all user I/O pins. Once the FPGA successfully exits POR, all user I/O pins are tri-stated. APEX 20KE and APEX 20KC devices have weak pull-up resistors on the user I/O pins which are on before and during configuration.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the Operating Conditions table of the appropriate device family data sheet.

The configuration cycle consists of 3 stages: reset, configuration and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration in this scheme, the microprocessor must generate a low-to-high transition on the nCONFIG pin.

VCCINT and VCCIO pins on the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. Once nSTATUS is released the FPGA is ready to receive configuration data and the configuration stage begins. When nSTATUS is pulled high, the microprocessor should place the configuration data one byte at a time on the DATA[7..0] pins. New configuration data should be sent to the FPGA every eight DCLK cycles.

The APEX 20KE or APEX 20KC device receives configuration data on its DATA[7..0] pins and the clock is received on the DCLK pin. On the first rising DCLK edge, a byte of configuration data is latched into the target device; the subsequent eight falling DCLK edges serialize the configuration data in the device. On the ninth rising clock edge, the next byte of configuration data is latched and serialized into the target device.

Data is clocked into the target device until CONF_DONE goes high. After the FPGA has received all configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by an external 10-kΩ pull-up resistor. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin.

In APEX 20KE and APEX 20KC devices, the initialization process is synchronous and can be clocked by its internal oscillator (typically 10 MHz) or by the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the APEX 20KE or APEX 20KC device will take care to provide itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device. Driving DCLK to the device after configuration is complete does not affect device operation.

You also have the flexibility to synchronize initialization of multiple devices by using the CLKUSR option. The *Enable user-supplied start-up clock (CLKUSR)* option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR will not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, APEX 20KE and APEX 20KC devices require 40 clock cycles to initialize properly.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. This *Enable INIT_DONE output* option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used it will be high due to an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low. When initialization is complete, the INIT_DONE pin will be released and pulled high. The microprocessor must be able to detect this low-to-high transition which signals the FPGA has entered user mode. In user-mode, the user I/O pins will no longer have weak pull-ups and will function as assigned in your design. When initialization is complete, the FPGA enters user mode.

To ensure DCLK and DATA0 are not left floating at the end of configuration, the microprocessor must take care to drive them either high or low, whichever is convenient on your board. The DATA[7..1] pins are available as user I/O pins after configuration. When the PPS scheme is chosen in the Quartus II software, as a default these I/O pins are tri-stated in user mode and should be driven by the microprocessor. To change this default option in the Quartus II software, select the **Dual-Purpose Pins** tab of the **Device & Pin Options** dialog box.

The configuration clock (DCLK) speed must be below the specified frequency, as listed in Table 7–5, to ensure correct configuration. No maximum DCLK period exists, which means you can pause configuration by halting DCLK for an indefinite amount of time. An optional status pin (RDYnBSY) on the FPGA indicates when it is busy serializing configuration data and when it is ready to accept the next data byte. The RDYnBSY pin is not required in the PPS mode. Configuration data can be sent every 8 DCLK cycles without monitoring this status pin.

If an error occurs during configuration, the FPGA drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the microprocessor that there is an error. If the *Auto-Restart Configuration on Error* option-available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box-is turned on, the FPGA releases nSTATUS after a reset time-out period (maximum of 40 µs). After nSTATUS is released and pulled high by a pull-up resistor, the microprocessor can try to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition (with a low pulse of at least 8 µs) on nCONFIG to restart the configuration process.

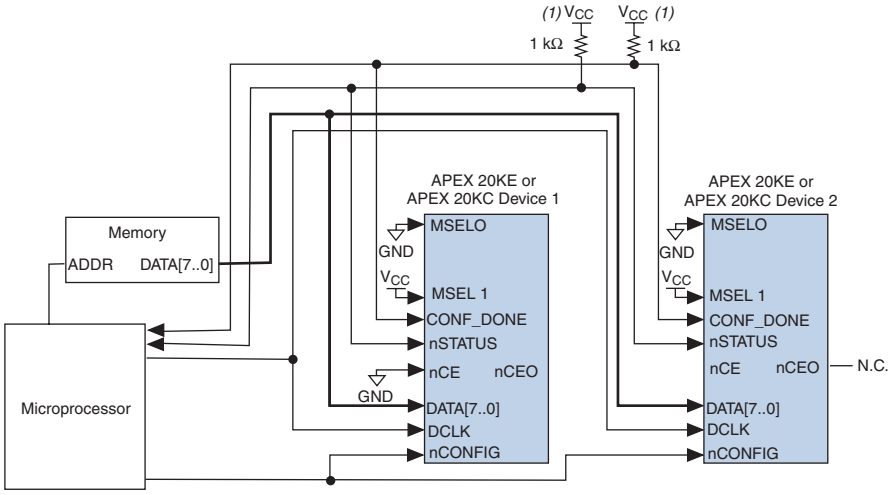
The microprocessor can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The CONF_DONE pin must be monitored by the microprocessor to detect errors and determine when programming completes. If the microprocessor sends all configuration data but CONF_DONE or INIT_DONE have not gone high, the microprocessor must reconfigure the target device.

If the optional CLKUSR pin is being used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure CLKUSR continues toggling during the time nSTATUS is low (maximum of 40 µs).

When the FPGA is in user-mode, a reconfiguration can be initiated by transitioning the nCONFIG pin low-to-high. The nCONFIG pin should be low for at least 8 µs for APEX 20KE and APEX 20KC devices. When nCONFIG is pulled low, the FPGA also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Once nCONFIG returns to a logic high state and nSTATUS is released by the FPGA, reconfiguration begins.

Figure 7–17 shows how to configure multiple APEX 20KE and APEX 20KC devices using a microprocessor. This circuit is similar to the PPS configuration circuit for a single device, except the devices are cascaded for multi-device configuration.

Figure 7–17. Multi-Device PPS Configuration Using a Microprocessor



Note to Figure 7–17:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.

In multi-device PPS configuration the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration within one clock cycle; therefore, the transfer of data destinations is transparent to the microprocessor.

Altera recommends keeping the configuration data valid on the DATA[7..0] bus for the 8 serializing clock cycles. The configuration data should be held valid on the DATA bus for the complete byte period because the nCEO of the first (and preceding) device can go low during the serializing DCLK cycles. Once the nCEO of the first (and preceding) device goes low, the second (and next) device becomes active and will begin trying to accept configuration data. If the configuration data is not valid on the first DCLK edge after nCEO goes low, then the second device will see incorrect configuration data and will never begin accepting configuration data. This situation will only arise if you are sharing the DATA[7..0] bus with other system data such that the configuration data is only valid for a portion of the byte period.

If your system requires to bus-share the DATA[7..0] line, you can work-around this by ensuring that the second (or next) device sees correct configuration data on the first rising edge of DCLK after the nCEO signal goes low. This can be achieved by delaying the nCEO signal by using external registers or by presenting the next byte of configuration data after the nCEO transition.

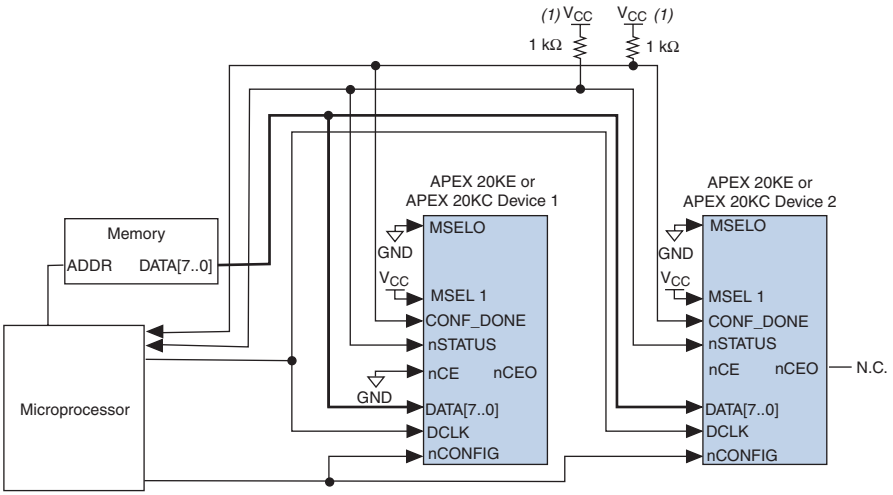
All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they may require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DCLK and DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Since all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first FPGA flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single FPGA detecting an error.

If the *Auto-Restart Configuration on Frame Error* option is turned on, the FPGAs release their nSTATUS pins after a reset time-out period (maximum of 40 µs). After all nSTATUS pins are released and pulled high, the microprocessor can try to reconfigure the chain without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition (with a low pulse of at least 8 µs) on nCONFIG to restart the configuration process.

In your system, you may have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they may require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete configuration at the same time. Figure 7–18 shows multi-device PPS configuration when both devices are receiving the same configuration data.

Figure 7–18. Multiple-Device PPS Configuration Using a Microprocessor When Both FPGAs Receive the Same Data



Notes to Figure 7–18:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.

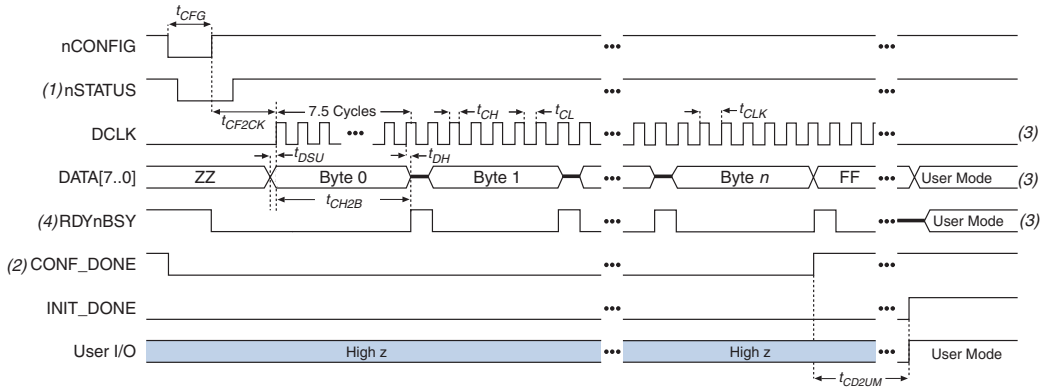
(2) The nCEO pins of both devices are left unconnected when configuring the same configuration data into multiple devices.

You can use a single configuration chain to configure APEX 20KE or APEX 20KC devices with other Altera devices that support PPS configuration, such as Mercury™, ACEX® 1K, or FLEX® 10K devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device `CONF_DONE` and `nSTATUS` pins must be tied together.

For more information on configuring multiple Altera devices in the same configuration chain, refer to *Configuring Mixed Altera FPGA Chains* in the *Configuration Handbook*.

Figure 7–19 shows the timing waveform for the PPS configuration scheme using a microprocessor.

Figure 7–19. APEX 20KE & APEX 20KC PPS Configuration Timing Waveform



Notes to Figure 7–19:

(1) Upon power-up, the APEX 20KE and APEX 20KC devices holds nSTATUS low for approximately 5 µs after VCC reaches its minimum requirement.

(2) Upon power-up, before and during configuration, CONF_DONE is low.

(3) DATA0 and DCLK should not be left floating after configuration. It should be driven high or low, whichever is more convenient. DATA[7..1] and RDYnBSY are available as user I/Os after configuration and the state of theses pins depends on the design programmed into the device.

(4) The RDYnBSY pin is not required in the PPS mode. Configuration data can be sent every 8 DCLK cycles without monitoring this status pin.

Table 7–5 defines the timing parameters for APEX 20KE and APEX 20KC devices for PPS configuration.

Table 7–5. PPS Timing Parameters for APEX 20KE & APEX 20KC Devices (Part 1 of 2)

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		200	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		200	ns
t_{CFG}	nCONFIG low pulse width	8		µs
t_{STATUS}	nSTATUS low pulse width	10	40 *(1)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high		1 *(1)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	40		µs
t_{ST2CK}	nSTATUS high to first rising edge on DCLK	1		µs
t_{DSU}	Data setup time before rising edge on DCLK	10		ns
t_{DH}	Data hold time after rising edge on DCLK	0		ns
t_{CH2B}	First rising DCLK to first rising RDYnBSY *(2)*	0.75 *(3)*		µs

Table 7–5. PPS Timing Parameters for APEX 20KE & APEX 20KC Devices (Part 2 of 2)

Symbol	Parameter	Min	Max	Units
t_{CH}	DCLK high time	15		ns
t_{CL}	DCLK low time	15		ns
t_{CLK}	DCLK period	30		ns
f_{MAX}	DCLK frequency		33.3	MHz
t_{CD2UM}	CONF_DONE high to user mode *(4)*	2	8	µs

Notes to Table 7–5:

(1) This value is obtainable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(2) The RDYnBSY pin is not required in the PPS mode. Configuration data can be sent every 8 DCLK cycles without monitoring this status pin.

(3) This parameter depends on the DCLK frequency. The RDYnBSY signal goes high 7.5 clock cycles after the rising edge of DCLK. This value was calculated with a DCLK frequency of 10 MHz.

(4) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device. If the clock source is CLKUSR, multiply the clock period by 40 to obtain this value.

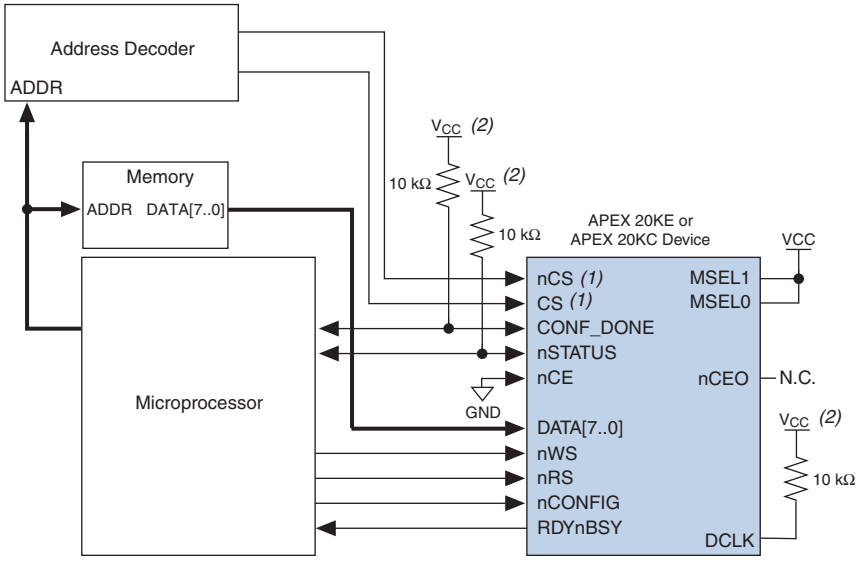
Device configuration options and how to create configuration files are discussed further in *Software Settings*, chapter 6 and 7 in volume, 2 of the *Configuration Handbook*.

Passive Parallel Asynchronous Configuration

Passive Parallel Asynchronous (PPA) configuration uses an intelligent host, such as a microprocessor, to transfer configuration data from a storage device, such as flash memory, to the target APEX 20KE or APEX 20KC device. Configuration data can be stored in TTF, RBF, or HEX format. The host system outputs byte-wide data and the accompanying strobe signals to the FPGA. When using PPA, you should pull the DCLK pin high through a 10-kΩ pull-up resistor to prevent unused configuration input pins from floating.

Figure 7–20 shows the configuration interface connections between the FPGA and a microprocessor for single device PPA configuration. The microprocessor or an optional address decoder can control the device's chip select pins, nCS and CS. The address decoder allows the microprocessor to select the APEX 20KE or APEX 20KC device by accessing a particular address, which simplifies the configuration process. The nCS and CS pins must be held active during configuration and initialization.

Figure 7–20. Single Device PPA Configuration Using a Microprocessor *Note (1)*



Notes to Figure 7–20:

(1) If not used, the CS pin can be connected to V_{CC} directly. If not used, the nCS pin can be connected to GND directly.

(2) The pull-up resistor should be connected to a supply that provides an acceptable input signal for the device.

During PPA configuration, it is only required to use either the nCS or CS pin. Therefore, if only one chip-select input is used, the other must be tied to the active state. For example, nCS can be tied to ground while CS is toggled to control configuration. The device's nCS or CS pins can be toggled during PPA configuration if the design meets the specifications set for t_{CSSU}, t_{WSP}, and t_{CSH} listed in Table 7–6.

Upon power-up, the APEX 20KE or APEX 20KC device goes through a Power-On Reset (POR) for approximately 5 µs. During POR, the device resets and holds nSTATUS low, and tri-states all user I/O pins. Once the FPGA successfully exits POR, all user I/O pins are tri-stated. APEX 20KE and APEX 20KC devices have weak pull-up resistors on the user I/O pins which are on before and during configuration.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the Operating Conditions table of the *APEX 20K Programmable Logic Device Family Data Sheet* or *APEX 20KC Programmable Logic Device Family Data Sheet*.

The configuration cycle consists of 3 stages: reset, configuration and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration, the microprocessor must generate a low-to-high transition on the nCONFIG pin.

VCCINT and VCCIO pins on the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. Once nSTATUS is released, the FPGA is ready to receive configuration data and the configuration stage begins. When nSTATUS is pulled high, the microprocessor should then assert the target device's nCS pin low and/or CS pin high. Next, the microprocessor places an 8-bit configuration word (one byte) on the target device's DATA[7..0] pins and pulses the nWS pin low.

On the rising edge of nWS, the target device latches in a byte of configuration data and drives its RDYnBSY signal low, which indicates it is processing the byte of configuration data. The microprocessor can then perform other system functions while the APEX 20KE or APEX 20KC device is processing the byte of configuration data.

During the time RDYnBSY is low, the APEX 20KE or APEX 20KC device internally processes the configuration data using its internal oscillator (typically 10 MHz). When the device is ready for the next byte of configuration data, it will drive RDYnBSY high. If the microprocessor senses a high signal when it polls RDYnBSY, the microprocessor sends the next byte of configuration data to the FPGA.

Alternatively, the nRS signal can be strobed low, causing the RDYnBSY signal to appear on DATA7. Because RDYnBSY does not need to be monitored, this pin doesn't need to be connected to the microprocessor. Data should not be driven onto the data bus while nRS is low because it will cause contention on the DATA7 pin. If the nRS pin is not used to monitor configuration, it should be tied high.

To simplify configuration and save an I/O port, the microprocessor can wait for the total time of t_{BUSY}(max) + t_{RDY2WS} + t_{W2SB} before sending the next data byte. In this set-up, nRS should be tied high and RDYnBSY does not need to be connected to the microprocessor. The t_{BUSY}, t_{RDY2WS} and t_{W2SB} timing specifications are listed in Table 7–6.

Next, the microprocessor checks nSTATUS and CONF_DONE. If nSTATUS is not low and CONF_DONE is not high, the microprocessor sends the next data byte. However, if nSTATUS is not low and all the configuration data has been received, the device is ready for initialization. After the FPGA has received all configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by an external 10-kΩ pull-up resistor. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin.

In APEX 20KE and APEX 20KC devices, the initialization clock source is either the APEX 20KE or APEX 20KC internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the APEX 20KE or APEX 20KC device will take care to provide itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device.

You also have the flexibility to synchronize initialization of multiple devices by using the CLKUSR option. The *Enable user-supplied start-up clock (CLKUSR)* option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR will not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, APEX 20KE and APEX 20KC devices require 40 clock cycles to initialize properly.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. This *Enable INIT_DONE output* option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used it will be high due to an external 10-kΩ pull-up when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low. When initialization is complete, the INIT_DONE pin will be released and pulled high. The microprocessor must be able to detect this low-to-high transition which signals the FPGA has entered user mode. In user-mode, the user I/O pins will no longer have weak pull-ups and will function as assigned in your design. When initialization is complete, the FPGA enters user mode.

To ensure DATA0 is not left floating at the end of configuration, the microprocessor must take care to drive them either high or low, whichever is convenient on your board. After configuration, the nCS, CS, nRS, nWS, RDYnBSY, and DATA[7..1] pins can be used as user I/O pins. When the PPA scheme is chosen in the Quartus II software, as a default these I/O pins are tri-stated in user mode and should be driven by the microprocessor. To change this default option in the Quartus II software, select the **Dual-Purpose Pins** tab of the **Device & Pin Options** dialog box.

If an error occurs during configuration, the FPGA drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the microprocessor that there is an error. If the *Auto-Restart Configuration After Error* option-available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box-is turned on, the FPGA releases nSTATUS after a reset time-out period (maximum of 40 µs). After nSTATUS is released and pulled high by a pull-up resistor, the microprocessor can try to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition (with a low pulse of at least 8 µs) on nCONFIG to restart the configuration process.

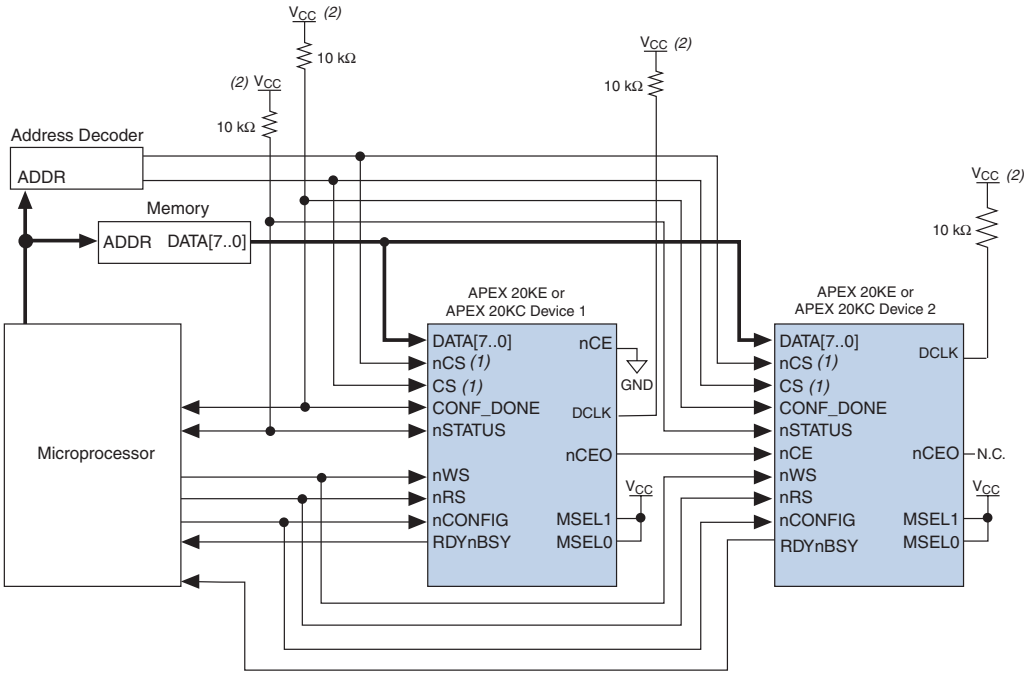
The microprocessor can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The CONF_DONE pin must be monitored by the microprocessor to detect errors and determine when programming completes. If the microprocessor sends all configuration data but CONF_DONE or INIT_DONE has not gone high, the microprocessor must reconfigure the target device.

If the optional CLKUSR pin is being used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure CLKUSR continues toggling during the time nSTATUS is low (maximum of 40 µs).

When the FPGA is in user-mode, a reconfiguration can be initiated by transitioning the nCONFIG pin low-to-high. The nCONFIG pin should be low for at least 8 µs. When nCONFIG is pulled low, the FPGA also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Once nCONFIG returns to a logic high state and nSTATUS is released by the FPGA, reconfiguration begins.

Figure 7–21 shows how to configure multiple APEX 20KE and APEX 20KC devices using a microprocessor. This circuit is similar to the PPA configuration circuit for a single device, except the devices are cascaded for multi-device configuration.

Figure 7–21. Multi-Device PPA Configuration Using a Microprocessor



Notes to Figure 7–21:

(1) If not used, the CS pin can be connected to V_{CC} directly. If not used, the nCS pin can be connected to GND directly.

(2) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.

In multi-device PPA configuration the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration within one clock cycle; therefore, the transfer of data destinations is transparent to the microprocessor.

Each device's RDYnBSY pin can have a separate input to the microprocessor. Alternatively, if the microprocessor is pin limited, all the RDYnBSY pins can feed an AND gate and the output of the AND gate can feed the microprocessor. For example, if you have 2 devices in a PPA configuration chain, the second device's RDYnBSY pin will be high during the time that the first device is being configured. When the first device has been successfully configured, it will driven nCEO low to activate the next device in the chain and drive its RDYnBSY pin high. Therefore, since RDYnBSY signal is driven high before configuration and after configuration before entering user-mode, the device being configured will govern the output of the AND gate.

The nRS signal can be used in multi-device PPA chain since the APEX 20KE or APEX 20KC device will tri-state its DATA[7..0] pins before configuration and after configuration before entering user-mode to avoid contention. Therefore, only the device that is currently being configured will respond to the nRS strobe by asserting DATA7.

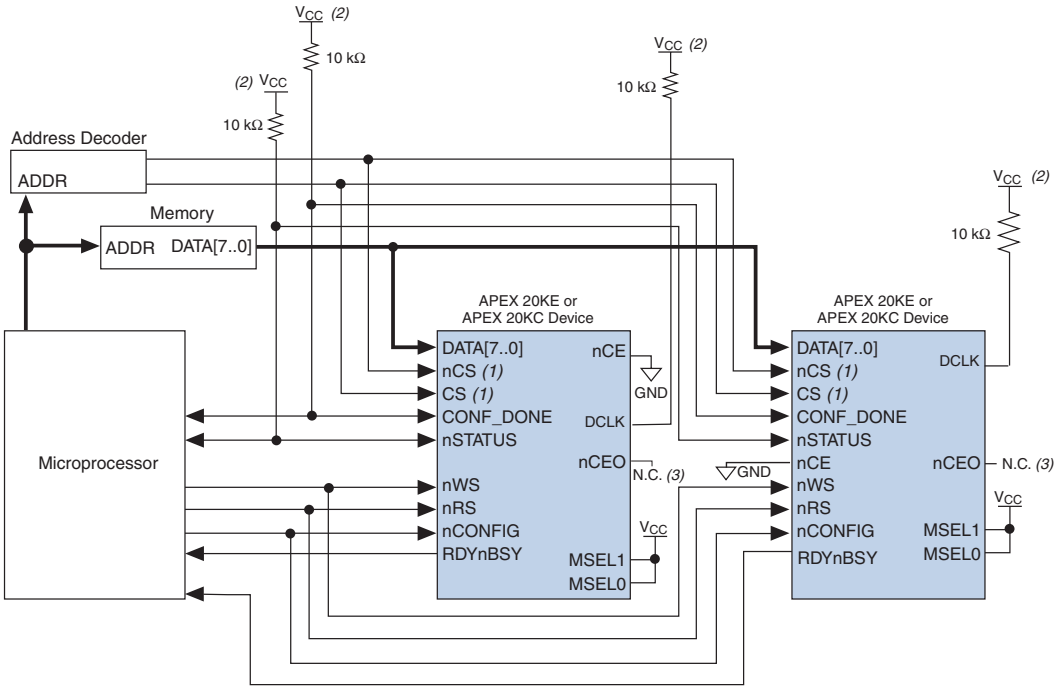
All other configuration pins (nCONFIG, nSTATUS, DATA[7..0], nCS, CS, nWS, nRS and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they may require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Since all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first FPGA flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single FPGA detecting an error.

If the *Auto-Restart Configuration After Error* option is turned on, the FPGAs release their nSTATUS pins after a reset time-out period (maximum of 40 µs). After all nSTATUS pins are released and pulled high, the microprocessor can try to reconfigure the chain without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition (with a low pulse of at least 8 µs) on nCONFIG to restart the configuration process.

In your system, you may have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DATA[7..1], nCS, CS, nWS, nRS and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they may require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete configuration at the same time. Figure 7–22 shows multi-device PPA configuration when both devices are receiving the same configuration data.

Figure 7–22. Multiple-Device PPA Configuration Using a Microprocessor When Both FPGAs Receive the Same Data



Notes to Figure 7–22:

(1) If not used, the CS pin can be connected to V_{CC} directly. If not used, the nCS pin can be connected to GND directly.

(2) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.

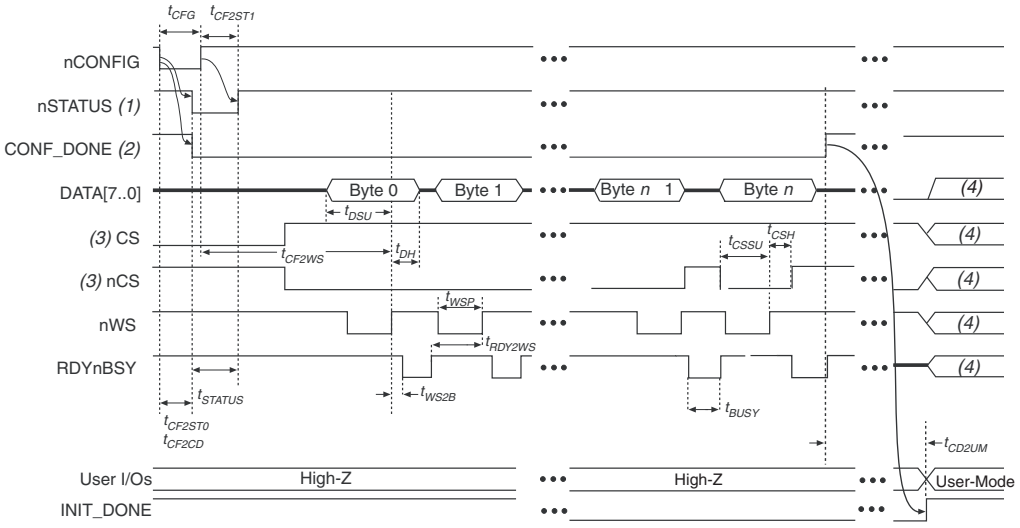
(3) The nCEO pins of both devices are left unconnected when configuring the same configuration data into multiple devices.

You can use a single configuration chain to configure APEX 20KE and APEX 20KC devices with other Altera devices that support PPA configuration, such as Stratix®, Mercury, APEX II, ACEX 1K, and FLEX 10KE devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information on configuring multiple Altera devices in the same configuration chain, refer to *Configuring Mixed Altera FPGA Chains* in the *Configuration Handbook*.

Figure 7–23 shows the timing waveform for the PPA configuration scheme using a microprocessor.

Figure 7–23. APEX 20KE & APEX 20KC PPA Configuration Timing Waveform



Notes to Figure 7–23:

(1) Upon power-up, the APEX 20KE or APEX 20KC device holds nSTATUS low for not more than 5 µs after V_{CCINT} reaches its minimum requirement.

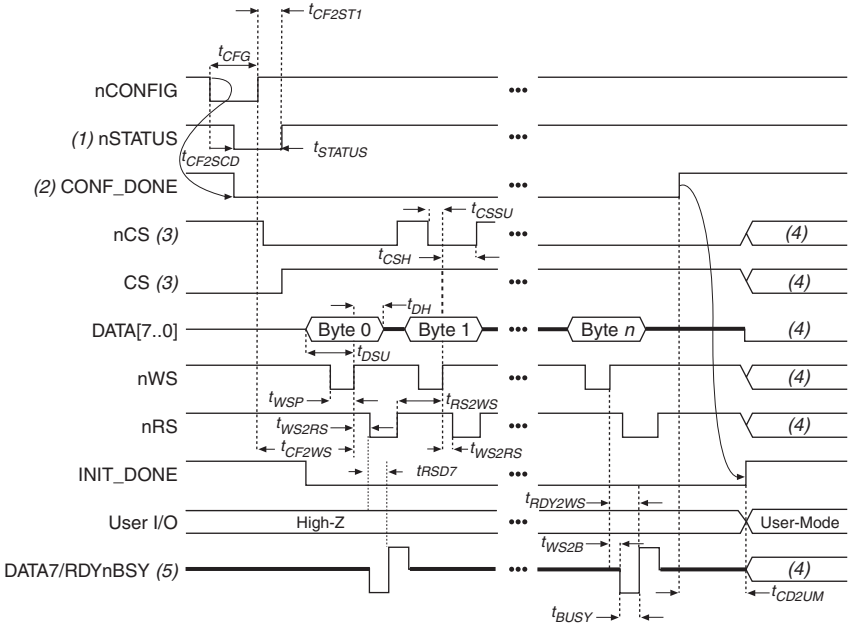
(2) Upon power-up, before and during configuration, CONF_DONE is low.

(3) The user can toggle nCS or CS during configuration if the design meets the specification for t_{CSSU}, t_{WSP}, and t_{CSH}.

(4) DATA0 should not be left floating after configuration. It should be driven high or low, whichever is more convenient. DATA[7..1], CS, nCS, nWS, nRS and RDYnBSY are available as user I/O pins after configuration and the state of theses pins depends on the dual-purpose pin settings.

Figure 7–24 shows the timing waveform for the PPA configuration scheme when using a strobed nRS and nWS signal.

Figure 7–24. APEX 20KE & APEX 20KC PPA Configuration Timing Waveform Using nRS & nWS



Notes to Figure 7–24:

(1) Upon power-up, the APEX 20KE or APEX 20KC device holds nSTATUS low for not more than 5 µs after V_{CCINT} reaches its minimum requirement.

(2) Upon power-up, before and during configuration, CONF_DONE is low.

(3) The user can toggle nCS or CS during configuration if the design meets the specification for t_{CSSU}, t_{WSP}, and t_{CSH}.

(4) DATA0 should not be left floating after configuration. It should be driven high or low, whichever is more convenient. DATA[7..1], CS, nCS, nWS, nRS, and RDYnBSY are available as user I/O pins after configuration and the state of theses pins depends on the dual-purpose pin settings.

(5) DATA7 is a bidirectional pin. It is an input for configuration data input, but it is an output to show the status of RDYnBSY.

Table 7–6 defines the timing parameters for APEX 20KE or APEX 20KC devices for PPA configuration.

Table 7–6. PPA Timing Parameters for APEX 20KE & APEX 20KC Devices (Part 1 of 2)

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		200	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		200	ns
t_{CFG}	nCONFIG low pulse width	8		µs
t_{STATUS}	nSTATUS low pulse width	10	40 *(1)*	µs

Table 7–6. PPA Timing Parameters for APEX 20KE & APEX 20KC Devices (Part 2 of 2)

Symbol	Parameter	Min	Max	Units
t_{CF2ST1}	nCONFIG high to nSTATUS high		1 *(1)*	µs
t_{CSSU}	Chip select setup time before rising edge on nWS	10		ns
t_{CSH}	Chip select hold time after rising edge on nWS	0		ns
t_{CF2WS}	nCONFIG high to first rising edge on nWS	40		µs
t_{DSU}	Data setup time before rising edge on nWS	10		ns
t_{DH}	Data hold time after rising edge on nWS	0		ns
t_{WSP}	nWS low pulse width	200		ns
t_{WS2B}	nWS rising edge to RDYnBSY low		50	ns
t_{BUSY}	RDYnBSY low pulse width	0.1	1.6	µs
t_{RDY2WS}	RDYnBSY rising edge to nWS rising edge	50		ns
t_{WS2RS}	nWS rising edge to nRS falling edge	200		ns
t_{RS2WS}	nRS rising edge to nWS rising edge	200		ns
t_{RSD7}	nRS falling edge to DATA7 valid with RDYnBSY signal		50	ns
t_{CD2UM}	CONF_DONE high to user mode *(2)*	2	8	µs

Notes to Table 7–6:

(1) This value is obtainable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(2) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device. If the clock source is CLKUSR, multiply the clock period by 40 for APEX 20KE and APEX 20KC devices to obtain this value.

Device configuration options and how to create configuration files are discussed further in *Software Settings*, chapter 6 and 7 in volume 2 of the *Configuration Handbook*.

JTAG Configuration

The Joint Test Action Group (JTAG) has developed a specification for boundary-scan testing. This boundary-scan test (BST) architecture offers the capability to efficiently test components on PCBs with tight lead spacing. The BST architecture can test pin connections without using physical test probes and capture functional data while a device is operating normally. The JTAG circuitry can also be used to shift configuration data into the device.

For more information on JTAG boundary-scan testing, refer to *Application Note 39: IEEE 1149.1 (JTAG) Boundary-Scan Testing in Altera Devices*.

A device operating in JTAG mode uses four required pins, TDI, TDO, TMS, and TCK, and one optional pin, TRST. All user I/O pins are tri-stated during JTAG configuration. APEX 20KE and APEX 20KC devices are designed such that JTAG instructions have precedence over any device configuration modes. This means that JTAG configuration can take place without waiting for other configuration modes to complete. For example, if you attempt JTAG configuration of APEX 20KE and APEX 20KC FPGAs during PS configuration, PS configuration will be terminated and JTAG configuration will begin.

Table 7–7 explains each JTAG pin's function.

Table 7–7. JTAG Pin Descriptions

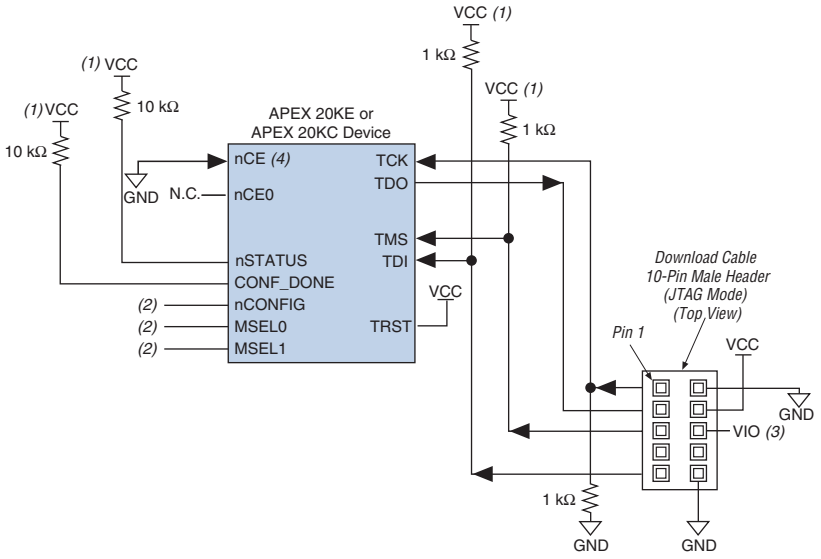
Pin	Description	Function
TDI	Test data input	Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}.
TDO	Test data output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by leaving this pin unconnected.
TMS	Test mode select	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}.
TCK	Test clock input	The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND.
TRST	Test reset input (optional)	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to IEEE Std. 1149.1. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND.

If V_{CCIO} of the bank where the JTAG pins reside, are tied to 3.3-V, both the I/O pins and JTAG TDO port will drive at 3.3-V levels.

During JTAG configuration, data can be downloaded to the device on the PCB through the USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV header. Configuring devices through a cable is similar to

programming devices in-system, except the TRST pin should be connected to V_{CC}. This ensures that the TAP controller is not reset. Figure 7–25. shows JTAG configuration of a single APEX 20KE or APEX 20KC device.

Figure 7–25. JTAG Configuration of a Single Device Using a Download Cable



Notes to Figure 7–25:

(1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster (VIO pin), ByteBlaster II, or ByteBlasterMV cable.

(2) The nCONFIG, MSEL0, and MSEL1 pins should be connected to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect nCONFIG to V_{CC}, and MSEL0 and MSEL1 to ground.

(3) Pin 6 of the header is a VIO reference voltage for the MasterBlaster output driver. VIO should match the device's V_{CCIO}. Refer to the MasterBlaster Serial/USB Communications Cable Data Sheet for this value. In the ByteBlasterMV cable, this pin is a no connect. In the USB Blaster and ByteBlaster II cable, this pin is connected to nCE when it is used for Active Serial programming, otherwise it is a no connect.

(4) nCE must be connected to GND or driven low for successful JTAG configuration.

To configure a single device in a JTAG chain, the programming software places all other devices in BYPASS mode. In BYPASS mode, devices pass programming data from the TDI pin to the TDO pin through a single bypass register without being affected internally. This scheme enables the programming software to program or verify the target device. Configuration data driven into the device appears on the TDO pin one clock cycle later.

APEX 20KE and APEX 20KC devices have dedicated JTAG pins that always function as JTAG pins. JTAG testing can be performed on APEX 20KE and APEX 20KC devices both before and after configuration, but not during configuration. The chip-wide reset (DEV_CLRn) and chip-wide output enable (DEV_OE) pins on APEX 20KE and APEX 20KC devices do not affect JTAG boundary-scan or programming operations. Toggling these pins does not affect JTAG operations (other than the usual boundary-scan operation).

When designing a board for JTAG configuration of APEX 20KE and APEX 20KC devices, the dedicated configuration pins should be considered. Table 7–8 shows how these pins should be connected during JTAG configuration.

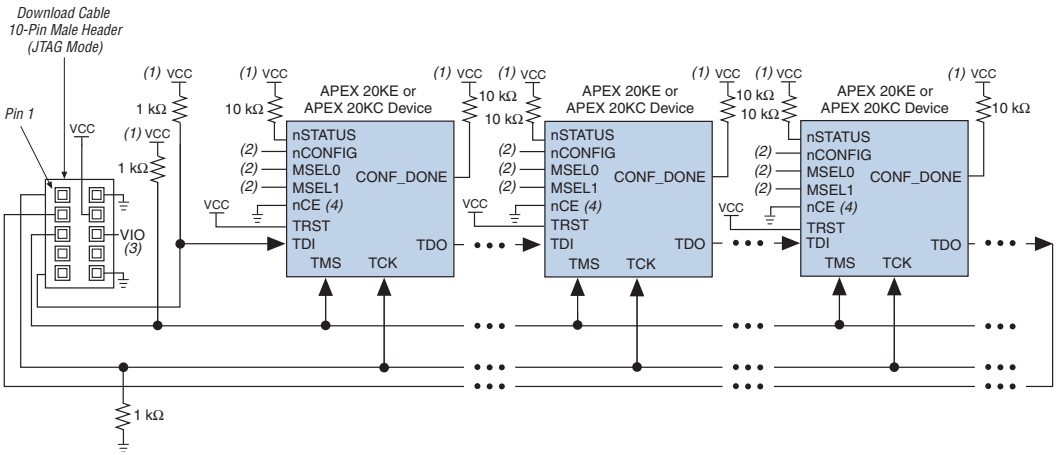
Table 7–8. Dedicated Configuration Pin Connections During JTAG Configuration

Signal	Description
nCE	On all APEX 20KE and APEX 20KC devices in the chain, nCE should be driven low by connecting it to ground, pulling it low via a resistor, or driving it by some control circuitry. For devices that are also in multi-device PS, PPS or PPA configuration chains, the nCE pins should be connected to GND during JTAG configuration or JTAG configured in the same order as the configuration chain.
nCEO	On all APEX 20KE and APEX 20KC devices in the chain, nCEO can be left floating or connected to the nCE of the next device. See nCE description above.
MSEL	These pins must not be left floating. These pins support whichever non-JTAG configuration is used in production. If only JTAG configuration is used, you should tie both pins to ground.
nCONFIG	Driven high by connecting to V_{CC}, pulling high via a resistor, or driven by some control circuitry.
nSTATUS	Pull to V_{CC} via a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each nSTATUS pin should be pulled up to V_{CC} individually. nSTATUS pulling low in the middle of JTAG configuration indicates that an error has occurred.
CONF_DONE	Pull to V_{CC} via a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each CONF_DONE pin should be pulled up to V_{CC} individually. CONF_DONE going high at the end of JTAG configuration indicates successful configuration.
DCLK	Should not be left floating. Drive low or high, whichever is more convenient on your board.
DATA0	Should not be left floating. Drive low or high, whichever is more convenient on your board.

When programming a JTAG device chain, one JTAG-compatible header is connected to several devices. The number of devices in the JTAG chain is limited only by the drive capability of the download cable. When four or more devices are connected in a JTAG chain, Altera recommends buffering the TCK, TDI, and TMS pins with an on-board buffer.

JTAG-chain device programming is ideal when the system contains multiple devices, or when testing your system using JTAG BST circuitry. Figure 7–26 shows multi-device JTAG configuration.

Figure 7–26. JTAG Configuration of Multiple Devices Using a Download Cable



Notes to Figure 7–26:

(1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster (VIO pin), ByteBlaster II or ByteBlasterMV cable.

(2) The nCONFIG, MSEL0, and MSEL1 pins should be connected to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect nCONFIG to V_{CC}, and MSEL0 and MSEL1 to ground.

(3) Pin 6 of the header is a VIO reference voltage for the MasterBlaster output driver. VIO should match the device's V_{CCIO}. Refer to the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value. In the ByteBlasterMV, this pin is a no connect. In the USB Blaster and ByteBlaster II, this pin is connected to nCE when it is used for Active Serial programming, otherwise it is a no connect.

(4) nCE must be connected to GND or driven low for successful JTAG configuration.

The nCE pin must be connected to GND or driven low during JTAG configuration. In multi-device PS, PPS, and PPA configuration chains, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. Therefore, if these devices are also in a JTAG chain, you should make sure the nCE pins are connected to GND during JTAG configuration or that the devices are JTAG configured in the same order as the configuration chain. As long as the devices are JTAG configured in the same order as the multi-device configuration chain, the nCEO of the previous device will drive nCE of the next device low when it has successfully been JTAG configured.

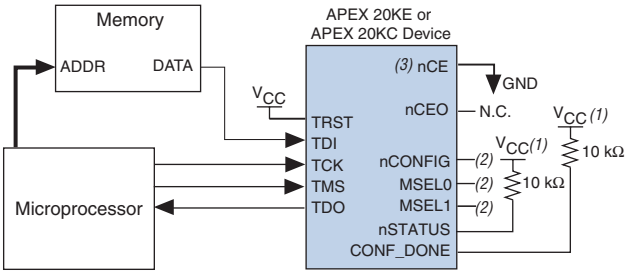
Other Altera devices that have JTAG support can be placed in the same JTAG chain for device programming and configuration.

For more information about configuring multiple Altera devices in the same configuration chain, refer to *Configuring Mixed Altera FPGA Chains* in the *Configuration Handbook*.

The Quartus II software verifies successful JTAG configuration upon completion. At the end of configuration, the software checks the state of CONF_DONE through the JTAG port. If CONF_DONE is not high, the Quartus II software indicates that configuration has failed. If CONF_DONE is high, the software indicates that configuration was successful. When Quartus II generates a Jam file for a multi-device chain, it contains instructions so that all the devices in the chain will be initialized at the same time.

Figure 7–27 shows JTAG configuration of an APEX 20KE or APEX 20KC FPGA with a microprocessor.

Figure 7–27. JTAG Configuration of a Single Device Using a Microprocessor



Notes to Figure 7–27:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.

(2) Connect the nCONFIG, MSEL1, and MSEL0 pins to support a non-JTAG configuration scheme. If your design only uses JTAG configuration, connect the nCONFIG pin to V_{CC} and the MSEL1 and MSEL0 pins to ground.

(3) nCE must be connected to GND or driven low for successful JTAG configuration.

Jam STAPL

Jam STAPL, JEDEC standard JESD-71, is a standard file format for in-system programmability (ISP) purposes. Jam STAPL supports programming or configuration of programmable devices and testing of electronic systems, using the IEEE 1149.1 JTAG interface. Jam STAPL is a freely licensed open standard.

The Jam Player provides an interface for manipulating the IEEE Std. 1149.1 JTAG TAP state machine.

For more information on JTAG and Jam STAPL in embedded environments, refer to *AN 122: Using Jam STAPL for ISP & ICR via an Embedded Processor*. To download the jam player, visit the Altera web site at:

www.altera.com/support/software/download/programming/jam/jam-index.jsp

Configuring APEX 20KE & APEX 20KC FPGAs with JRunner

JRunner is a software driver that allows you to configure Altera FPGAs, including APEX 20KE and APEX 20KC FPGAs, through the ByteBlaster II or ByteBlasterMV cables in JTAG mode. The programming input file supported is in RBF format. JRunner also requires a Chain Description File (**.cdf**) generated by the Quartus II software. JRunner is targeted for embedded JTAG configuration. The source code has been developed for the Windows NT operating system (OS). You can customize the code to make it run on other platforms.

For more information on the JRunner software driver, refer to the *JRunner Software Driver: An Embedded Solution to the JTAG Configuration White Paper* and the source files.

Device Configuration Pins

The following tables describe the connections and functionality of all the configuration related pins on the APEX 20KE or APEX 20KC device. Table 7–9 describes the dedicated configuration pins, which are required to be connected properly on your board for successful configuration. Some of these pins may not be required for your configuration schemes.

Table 7–9. Dedicated Configuration Pins on the APEX 20KE & APEX 20KC Device (Part 1 of 5)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
MSEL0 MSEL1	N/A	All	Input	Two-bit configuration input that sets the APEX 20KE or APEX 20KC device configuration scheme. See Table 7–3 for the appropriate connections. These pins must remain at a valid state during power-up, before nCONFIG is pulled low to initiate a reconfiguration and during configuration.
nCONFIG	N/A	All	Input	Configuration control input. Pulling this pin low during user-mode will cause the FPGA to lose its configuration data, enter a reset state, tri-state all I/O pins, and returning this pin to a logic high level will initiate a reconfiguration. If your configuration scheme uses an enhanced configuration device or EPC2 device, nCONFIG can be tied directly to V_{CC} or to the configuration device's nINIT_CONF pin. For successful configuration of APEX 20KE devices, nCONFIG must be tied to V_{CCINT} through a 10-kΩ pull-up resistor.

Table 7–9. Dedicated Configuration Pins on the APEX 20KE & APEX 20KC Device (Part 2 of 5)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nSTATUS	N/A	All	Bidirectional open-drain	The FPGA drives nSTATUS low immediately after power-up and releases it within 5 µs. (When using a configuration device, the configuration device holds nSTATUS low for up to 200 ms.) Status output. If an error occurs during configuration, nSTATUS is pulled low by the target device. Status input. If an external source drives the nSTATUS pin low during configuration or initialization, the target device enters an error state. Driving nSTATUS low after configuration and initialization does not affect the configured device. If a configuration device is used, driving nSTATUS low will cause the configuration device to attempt to configure the FPGA, but since the FPGA ignores transitions on nSTATUS in user-mode, the FPGA will not reconfigure. To initiate a reconfiguration, nCONFIG must be pulled low. The enhanced configuration devices' and EPC2 devices' OE and nCS pins have optional internal programmable pull-up resistors. For successful configuration of APEX 20KE and APEX 20KC devices using EPC2 devices, use an external 10-kΩ pull-up resistor. If internal pull-up resistors on the enhanced configuration devices are used, external 10-kΩ pull-up resistors should not be used on these pins.
CONF_DONE	N/A	All	Bidirectional open-drain	Status output. The target FPGA drives the CONF_DONE pin low before and during configuration. Once all configuration data is received without error and the initialization cycle starts, the target device releases CONF_DONE. Status input. After all data is received and CONF_DONE goes high, the target device initializes and enters user mode. Driving CONF_DONE low after configuration and initialization does not affect the configured device. The enhanced configuration devices' and EPC2 devices' OE and nCS pins have optional internal programmable pull-up resistors. For successful configuration of APEX 20KE and APEX 20KC devices using EPC2 devices, use an external 10-kΩ pull-up resistor. If internal pull-up resistors on the enhanced configuration devices are used, external 10-kΩ pull-up resistors should not be used on these pins.

Table 7–9. Dedicated Configuration Pins on the APEX 20KE & APEX 20KC Device (Part 3 of 5)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nCE	N/A	All	Input	Active-low chip enable. The nCE pin activates the device with a low signal to allow configuration. The nCE pin must be held low during configuration, initialization, and user mode. In single device configuration, it should be tied low. In multi-device configuration, nCE of the first device is tied low while its nCEO pin is connected to nCE of the next device in the chain. The nCE pin must also be held low for successful JTAG programming of the FPGA.
nCEO	N/A	All	Output	Output that drives low when device configuration is complete. In single device configuration, this pin is left floating. In multi-device configuration, this pin feeds the next device's nCE pin. The nCEO of the last device in the chain is left floating.
DCLK	N/A	Synchronous configuration schemes (PS and PPS)	Input	Clock input used to clock data from an external source into the target device. Data is latched into the FPGA on the rising edge of DCLK. In PPA mode, DCLK should be tied high to V_{CC} to prevent this pin from floating. After configuration, this pin is tri-stated. In schemes that use a configuration device, DCLK will be driven low after configuration is done. In schemes that use a control host, DCLK should be driven either high or low, whichever is more convenient. Toggling this pin after configuration does not affect the configured device.
DATA0	N/A	All	Input	Data input. In serial configuration modes, bit-wide configuration data is presented to the target device on the DATA0 pin. After configuration, EPC1 and EPC1441 devices tri-state this pin, while enhanced and EPC2 devices drive this pin high. In schemes that use a control host, DATA0 should be driven either high or low, whichever is more convenient. Toggling this pin after configuration does not affect the configured device.
DATA[7..1]	I/O	Parallel configuration schemes (PPS and PPA)	Inputs	Data inputs. Byte-wide configuration data is presented to the target device on DATA[7..0]. In serial configuration schemes, they function as user I/O pins during configuration, which means they are tri-stated. After PPA or PPS configuration, DATA[7..1] are available as a user I/O pins and the state of these pin depends on the Dual-Purpose Pin settings.

Table 7–9. Dedicated Configuration Pins on the APEX 20KE & APEX 20KC Device (Part 4 of 5)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
DATA7	I/O	PPA	Bidirectional	In the PPA configuration scheme, the DATA7 pin presents the RDYnBSY signal after the nRS signal has been strobed low. In serial configuration schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, DATA7 is available as a user I/O and the state of this pin depends on the Dual-Purpose Pin settings.
nWS	I/O	PPA	Input	Write strobe input. A low-to-high transition causes the device to latch a byte of data on the DATA[7..0] pins. In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, nWS is available as a user I/O and the state of this pin depends on the Dual-Purpose Pin settings.
nRS	I/O	PPA	Input	Read strobe input. A low input directs the device to drive the RDYnBSY signal on the DATA7 pin. If the nRS pin is not used in PPA mode, it should be tied high. In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, nRS is available as a user I/O and the state of this pin depends on the Dual-Purpose Pin settings.

Table 7–9. Dedicated Configuration Pins on the APEX 20KE & APEX 20KC Device (Part 5 of 5)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
RDYnBSY	I/O	PPS and PPA	Output	Ready output. A high output indicates that the target device is ready to accept another data byte. A low output indicates that the target device is busy and not ready to receive another data byte. In PPS and PPA configuration schemes, this pin will drive out high after power-up, before configuration and after configuration before entering user-mode. In non-PPS and non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPS and PPA configuration, RDYnBSY is available as a user I/O and the state of this pin depends on the Dual-Purpose Pin settings.
nCS/CS	I/O	PPA	Input	Chip-select inputs. A low on nCS and a high on CS select the target device for configuration. The nCS and CS pins must be held active during configuration and initialization. During the PPA configuration mode, it is only required to use either the nCS or CS pin. Therefore, if only one chip-select input is used, the other must be tied to the active state. For example, nCS can be tied to ground while CS is toggled to control configuration. In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, nCS and CS are available as a user I/O pins and the state of these pins depends on the dual-purpose pin settings.

Table 7–10 describes the optional configuration pins. If these optional configuration pins are not enabled in the Quartus II software, they are available as general-purpose user I/O pins. Therefore during configuration they function as user I/O pins, which means they are tri-stated with weak pull-up resistors.

Table 7–10. Optional Configuration Pins

Pin Name	User Mode	Pin Type	Description
CLKUSR	N/A if option is on. I/O if option is off.	Input	Optional user-supplied clock input. Synchronizes the initialization of one or more devices. This pin is enabled by turning on the *Enable user-supplied start-up clock (CLKUSR)* option in the Quartus II software
INIT_DONE	N/A if option is on. I/O if option is off.	Output open-drain	Status pin. Can be used to indicate when the device has initialized and is in user mode. When nCONFIG is low and during the beginning of configuration, the INIT_DONE pin is tri-stated and pulled high due to an external 10-kΩ pull-up. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low. When initialization is complete, the INIT_DONE pin will be released and pulled high and the FPGA enters user mode. Thus, the monitoring circuitry must be able to detect a low-to-high transition. This pin is enabled by turning on the *Enable INIT_DONE output* option in the Quartus II software.
DEV_OE	N/A if option is on. I/O if option is off.	Input	Optional pin that allows the user to override all tri-states on the device. When this pin is driven low, all I/O pins are tri-stated; when this pin is driven high, all I/O pins behave as programmed. This pin is enabled by turning on the *Enable device-wide output enable (DEV_OE)* option in the Quartus II software.
DEV_CLRn	N/A if option is on. I/O if option is off.	Input	Optional pin that allows you to override all clears on all device registers. When this pin is driven low, all registers are cleared; when this pin is driven high, all registers behave as programmed. This pin is enabled by turning on the *Enable device-wide reset (DEV_CLRn)* option in the Quartus II software.

JTAG pins must be kept stable before and during configuration. JTAG pin stability prevents accidental loading of JTAG instructions. Table 7–11 describes the dedicated JTAG pins.

Table 7–11. Dedicated JTAG Pins

Pin Name	User Mode	Pin Type	Description
TDI	N/A	Input	Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}.
TDO	N/A	Output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by leaving this pin unconnected.
TMS	N/A	Input	Input pin that provides the control signal to determine the transitions of the TAP controller state machine.Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}.
TCK	N/A	Input	The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND.
TRST	N/A	Input	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to IEEE Std. 1149.1. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND.

APEX 20KE Power Sequencing

The following guidelines explain how to manage device power sequencing for APEX 20KE devices. These guidelines apply to all configuration schemes.

Altera has enhanced the APEX II and APEX 20KC devices, so you do not need to follow these guidelines for those devices. A system designed for an APEX 20KE device can successfully configure an APEX II or APEX 20KC device.

The APEX 20KE logic array and I/O pins can operate on different power supplies. V_{CCINT} powers the logic array, and each I/O bank has a separate V_{CCIO} supply.

These guidelines will allow you to configure APEX 20KE devices upon power-up as well as recover from power brown-out conditions. Specifically, V_{CCINT} can decrease to any voltage, and the device will successfully reconfigure when power is restored. During the brown-out condition, APEX 20KE devices may lose configuration if V_{CCINT} falls below the minimum V_{CCINT} device specification. If V_{CCINT} drops below the specified operating range, the APEX 20KE device resets and drives nSTATUS low. When V_{CCINT} is in the specified operating range, nSTATUS will be released and configuration will begin.

Stratix, Stratix GX, Cyclone, APEX II, APEX 20KC, Mercury, ACEX 1K, FLEX 10K, and FLEX 6000 devices' power supplies (V_{CCINT} and V_{CCIO}) can be powered up in either order.

When configuring APEX 20KE devices in the same configuration chain as other Altera FPGAs, you must follow these guidelines.

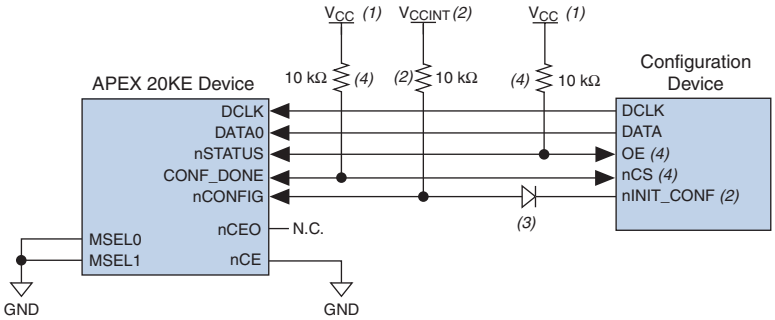
Power Sequencing Considerations

If the APEX 20KE device is configured by an external host, such as a microprocessor, ensure that the configuration controller holds nCONFIG low during power-up and then drives nCONFIG high to begin configuration when all power supplies are stable. This applies for all possible power-up sequences. External pull-ups used on nCONFIG, nSTATUS, and CONF_DONE should be 10 kΩ

To ensure recovery from brown-out conditions and successful configuration between APEX 20KE devices and configuration devices in all possible power-up sequences, pull nCONFIG up to V_{CCINT} through a 10-kΩ resistor and hold nCONFIG low for the entire time that the two supplies are powering up to their specified operating ranges. All other external pull-up resistors used on nSTATUS, and CONF_DONE should also be 10 kΩ

When using enhanced configuration devices or EPC2 device, nCONFIG of the FPGA can be connected to nINIT_CONF, which allows the INIT_CONF JTAG instruction to initiate FPGA configuration. Figure 7–28 shows the board connections used to support the initiate configuration JTAG instruction with the nINIT_CONF pin.

Figure 7–28. Configuring an APEX 20KE Device Using a Configuration Device



Notes to Figure 7–28:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CCINT} through a 10-kΩ resistor.

(3) The nINIT_CONF pin has an internal pull-up resistor to 3.3 V that is always active. Since a 10-kΩ pull-up to V_{CCINT} is required to successfully configure APEX 20KE devices, you need to isolate the 1.8-V V_{CCINT} from the configuration device's 3.3-V supply. To isolate the 1.8-V and 3.3-V power supplies, add a diode between the APEX 20KE device's nCONFIG pin and the configuration device's nINIT_CONF pin. Select a diode with a threshold voltage (V_T) less than or equal to 0.7 V. The diode will make the nINIT_CONF pin an open-drain pin; the pin will only be able to drive low or tri-state. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), this diode is not needed.

(4) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. For successful configuration of APEX 20KE devices, you should use external 10-kΩ pull-up resistors. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the *Disable nCS and OE pull-ups on configuration device* option when generating programming files.

The nINIT_CONF pin is an open-drain output and has an internal pull-up resistor to 3.3-V that is always active. Since a 10-kΩ pull-up to V_{CCINT} is required to successfully configure APEX 20KE devices, you need to isolate the 1.8-V V_{CCINT} from the configuration device's 3.3-V supply. To isolate the 1.8-V and 3.3-V power supplies, add a diode between the APEX 20KE device's nCONFIG pin and the configuration device's nINIT_CONF pin. Select a diode with a threshold voltage (V_T) less than or equal to 0.7 V. The diode will make the nINIT_CONF pin an open-drain pin; the pin will only be able to drive low or tri-state.

If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CCINT} through a 10-kΩ resistor and the isolating diode is not needed.

Use these guidelines to ensure a successful power-up and configuration:

- V_{CCINT} is powered before V_{CCIO}—These guidelines should be followed for applications where the sequence of the power supplies is unknown. It is highly recommended that you follow these guidelines to ensure successful configuration of your APEX 20KE device.
- V_{CCINT} is powered after V_{CCIO}—These guidelines should be followed if V_{CCIO} is powered before V_{CCINT}.

V_{CCINT} is Powered Before V_{CCIO}

If V_{CCINT} is powered before V_{CCIO}, the nCONFIG signal must be held low for the entire time that the two supplies are powering up to their specified operating ranges. The recommendations in this section should also be followed for applications where the sequence of the power supplies is unknown (e.g., hot-socketing applications). These guidelines should be followed for all configuration schemes using a configuration device or an external host, such as a microprocessor.

For more details on APEX 20KE and configuration device voltage supply specifications, refer to the *APEX 20K Programmable Logic Device Family Data Sheet* and the *Configuration Devices for Altera FPGAs Data Sheet* or *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet*, respectively.

Use one of the following methods to hold nCONFIG low:

- Use a power-monitoring circuit on the board. This circuit will drive nCONFIG low when V_{CCINT} and V_{CCIO} are out of range, and then release nCONFIG. nCONFIG can then be externally pulled up when V_{CCINT} returns to a normal operating level. National Semiconductor offers a small power-on reset circuit (part number LP3470) for a power monitor.
- Many voltage regulators offer a power-good signal that can be used to hold nCONFIG low during power-up. If there are multiple regulators, use the power-good signal on the regulator that powers up last. If the power sequence is unknown, the power-good signals can be ANDed in a discrete device.
- A microcontroller or intelligent host can externally drive nCONFIG low while both the V_{CCINT} and V_{CCIO} supplies are being powered.

V_{CCINT} is Powered After V_{CCIO}

If the APEX 20KE device is configured by an external host, such as a microprocessor, ensure that the configuration controller holds nCONFIG low during power-up and then drives nCONFIG high when all power supplies are stable to begin of configuration.

If the APEX 20KE device is configured using a configuration device and V_{CCINT} is powered up during or after the configuration device has exited POR (released nSTATUS/OE signal), nCONFIG must be tied to V_{CCINT} through a 10-kΩ resistor. The configuration device will exit POR 200 ms (maximum) after power-up.

When using an enhanced configuration device to configure any Altera FPGAs, including APEX 20KE devices, the V_{CCINT} of the FPGA must be powered before the enhanced configuration device exits POR (signaled by the low-to-high transition on the OE/nSTATUS signal). Power up needs to be controlled so that the enhanced configuration device's OE signal goes high after the CONF_DONE signal is pulled low.

If the FPGA is not powered up after the enhanced configuration device exits POR, the CONF_DONE signal will be high since the pull-up resistor is pulling this signal high. When the enhanced configuration device exits POR, OE is released and pulled high by a pull-up resistor. If the enhanced configuration device sees its nCS/CONF_DONE signal also high, the configuration device will go into an idle state because it sees the FPGA is already configured. DATA and DCLK will not toggle and the enhanced configuration device will only exit this idle state if it is powered down and then powered up correctly.

To ensure the enhanced configuration device enters configuration mode properly, you need to ensure that the FPGA completes its power-up before the enhanced configuration device exits POR. Alternatively the nSTATUS/OE line can be held low by an open-drain signal until after the V_{CCINT} power supply is stable.

For more information about power sequencing of enhanced configuration devices, refer to the *Enhanced Configuration Devices (EPC4, EPC8, & EPC16) Data Sheet* in the *Configuration Handbook*.

8. Configuring Mercury, APEX 20K (2.5 V), ACEX 1K & FLEX 10K Devices

CF51006-2.1

Introduction

Mercury™, APEX™ 20K (2.5 V), ACEX® 1K, and FLEX® 10K devices can be configured using one of four configuration schemes. All configuration schemes use either a microprocessor or configuration device.

This section covers how to configure Mercury, APEX 20K (2.5 V), ACEX 1K and FLEX 10K devices. APEX 20K (non-E and non-C) devices use a 2.5-V voltage supply for V_{CCINT}, while APEX 20KE and APEX 20KC devices use a 1.8-V voltage supply for V_{CCINT}. If your target FPGA is an APEX 20K device which uses a 1.8-V V_{CCINT},. For more information, refer to *Configuring APEX 20KE & APEX 20KC Devices* in the *Configuration Handbook*.

Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices can be configured using the passive serial (PS), passive parallel synchronous (PPS), passive parallel asynchronous (PPA), and Joint Test Action Group (JTAG) configuration schemes. The configuration scheme used is selected by driving the Mercury, APEX 20K (2.5 V), ACEX 1K, or FLEX 10K device `MSEL1` and `MSEL0` pins either high or low as shown in Table 8–1. If your application only requires a single configuration mode, the `MSEL` pins can be connected to V_{CC} (V_{CCIO} of the I/O bank where the `MSEL` pin resides) or to ground. If your application requires more than one configuration mode, you can switch the `MSEL` pins after the FPGA is configured

successfully. Toggling these pins during user-mode does not affect the device operation; however, the MSEL pins must be valid before a reconfiguration is initiated.

Table 8–1. Mercury, APEX 20K (2.5 V), ACEX 1K & FLEX 10K Configuration Schemes

MSEL1	MSEL0	Configuration Scheme
0	0	PS
1	0	PPS
1	1	PPA
(1)	*(1)*	JTAG Based *(2)*

Notes to Table 8–1:

(1) Do not leave the MSEL pins floating; connect them to a low- or high-logic level. These pins support the non-JTAG configuration scheme used in production. If only JTAG configuration is used, you should connect the MSEL pins to ground.

(2) JTAG-based configuration takes precedence over other configuration schemes, which means MSEL pin settings are ignored.

Tables 8–2 through 8–5 show the approximate configuration file sizes for Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices.

Table 8–2. Mercury Raw Binary File (.rbf) Sizes

Device	Data Size (Bits)	Data Size (Bytes)
EP1M120	1,303,120	162,890
EP1M350	4,394,032	549,254

Table 8–3. APEX 20K (2.5 V) Raw Binary File (.rbf) Sizes

Devices	Data Size (Bits)	Data Size (Bytes)
EP20K100	993,360	124,170
EP20K200	1,950,800	243,850
EP20K400	3,880,720	485,090

Table 8–4. ACEX 1K Raw Binary File (.rbf) Sizes

Devices	Data Size (Bits)	Data Size (Bytes)
EP1K10	159,160	19,895
EP1K30	473,720	59,215
EP1K50	784,184	98,023
EP1K100	1,335,720	166,965

Table 8–5. FLEX 10K Raw Binary File (.rbf) Sizes

Devices	Data Size (Bits)	Data Size (Bytes)
EPF10K30E	473,720	59,215
EPF10K50E	784,184	98,023
EPF10K50S	784,184	98,023
EPF10K100B	1,200,000	150,000
EPF10K100E	1,335,720	166,965
EPF10K130E	1,838,360	229,795
EPF10K200E	2,756,296	344,537
EPF10K200S	2,756,296	344,537
EPF10K10A	120,000	15,000
EPF10K30A	406,000	50,750
EPF10K50V	621,000	77,625
EPF10K100A	1,200,000	150,000
EPF10K130V	1,600,000	200,000
EPF10K250A	3,300,000	412,500
EPF10K10	118,000	14,750
EPF10K20	231,000	28,875
EPF10K30	376,000	47,000
EPF10K40	498,000	62,250
EPF10K50	621,000	77,625
EPF10K70	892,000	111,500
EPF10K100	1,200,000	150,000

Use the data in Tables 8–2 through 8–5 only to estimate the file size before design compilation. Different configuration file formats, such as a Hexidecimal (**.hex**) or Tabular Text File (**.ttf**) format, will have different file sizes. However, for any specific version of the Quartus® II or MAX+PLUS® II software, all designs targeted for the same device will have the same configuration file size.

The following chapter describes in detail how to configure Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices using the supported configuration schemes. The last section describes the device configuration pins available. In this chapter, the generic term device(s) or FPGA(s) will include all Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices.

For more information on setting device configuration options or creating configuration files, Refer to *Software Settings*, chapter 6 and 7 in volume 2 of the *Configuration Handbook*.

Passive Serial Configuration

You can perform Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K PS configuration using an Altera configuration device, an intelligent host (e.g., a microprocessor or Altera® MAX® device), or a download cable.

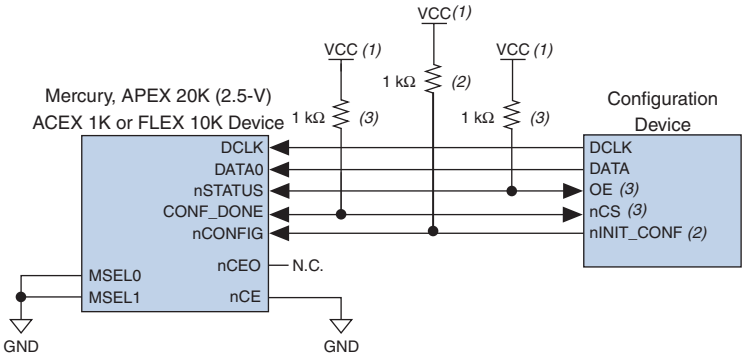
PS Configuration Using a Configuration Device

You can use an Altera configuration device, such as an enhanced configuration device, EPC2, or EPC1 device, to configure Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices using a serial configuration bitstream. Configuration data is stored in the configuration device. Figure 8–1 shows the configuration interface connections between Mercury, APEX 20K (2.5 V), ACEX 1K, or FLEX 10K devices and a configuration device.

The figures in this chapter only show the configuration-related pins and the configuration pin connections between the configuration device and the FPGA.

For more information on the configuration device and flash interface pins (e.g., PGM[2..0], EXCLK, PORSEL, A[20..0], and DQ[15..0]), refer to the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet*, chapter 2 in volume 1 of the *Configuration Handbook*.

Figure 8–1. Single Device PS Configuration Using a Configuration Device



Notes to Figure 8–1:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) has an internal pull-up resistor that is always active, meaning an external pull-up resistor is not required on the nINIT_CONF/nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the *Disable nCS and OE pull-ups on configuration device* option when generating programming files.

The value of the internal pull-up resistors on the enhanced configuration devices and EPC2 devices can be found in the Operating Conditions table of the *Enhanced Configuration Devices (EPC4, EPC8, & EPC16) Data Sheet* or the *Configuration Devices for SRAM-based LUT Devices Data Sheet* in the *Configuration Handbook*.

When using enhanced configuration devices or EPC2 devices, nCONFIG of the FPGA can be connected to nINIT_CONF, which allows the INIT_CONF JTAG instruction to initiate FPGA configuration. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CC} either directly or through a resistor. An internal pull-up on the nINIT_CONF pin is always active in enhanced configuration devices and EPC2 devices, which means an external pull-up is not required if nCONFIG is tied to nINIT_CONF.

Upon power-up, the Mercury, APEX 20K (2.5 V), ACEX 1K, or FLEX 10K device goes through a Power-On Reset (POR) for approximately 5 µs. During POR, the device resets and holds nSTATUS low, and tri-states all user I/O pins. The configuration device also goes through a POR delay to allow the power supply to stabilize. The POR time for EPC2, EPC1, and EPC1441 devices is 200 ms (maximum), and for enhanced configuration devices, the POR time can be set to either 100 ms or 2 ms, depending on its PORSEL pin setting. If the PORSEL pin is connected to GND, the POR delay is 100 ms. During this time, the configuration device drives its OE pin low. This low signal delays configuration because the OE pin is connected to the target device's nSTATUS pin. When both devices complete POR, they release their open-drain OE or nSTATUS pin, which is then pulled high by a pull-up resistor. Once the FPGA successfully exits POR, all user I/O pins are tri-stated. Mercury, APEX 20K (2.5 V), ACEX 1K and FLEX 10KE devices have weak pull-up resistors on the user I/O pins which are on before and during configuration.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the Operating Conditions table of the appropriate device family data sheet.

When the power supplies have reached the appropriate operating voltages, the target FPGA senses the low-to-high transition on nCONFIG and initiates the configuration cycle. The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. The beginning of configuration can be delayed by holding the nCONFIG or nSTATUS pin low.

VCCINT and VCCIO pins on the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the nSTATUS pin, which is pulled high by a pull-up resistor. Enhanced configuration and EPC2 devices have an optional internal pull-up on the OE pin. This option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If this internal pull-up resistor is not used, an external 1-kΩ pull-up resistor on the OE/nSTATUS line is required. Once nSTATUS is released, the FPGA is ready to receive configuration data and the configuration stage begins.

When nSTATUS is pulled high, OE of the configuration device also goes high and the configuration device clocks data out serially to the FPGA using its internal oscillator. The Mercury, APEX 20K (2.5 V), ACEX 1K, or FLEX 10K device receives configuration data on its DATA0 pin and the clock is received on the DCLK pin. Data is latched into the FPGA on the rising edge of DCLK.

After the FPGA has received all configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by a pull-up resistor. Since CONF_DONE is tied to the configuration device's nCS pin, the configuration device is disabled when CONF_DONE goes high. Enhanced configuration and EPC2 devices have an optional internal pull-up resistor on the nCS pin. This option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If this internal pull-up is not used, an external 1-kΩpull-up resistor on the nCS/CONF_DONE line is required. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin.

In Mercury and APEX 20K (2.5 V) devices, the initialization clock source is either the FPGA's internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Mercury or APEX 20K (2.5 V) device will allow enough clock cycles for proper initialization.

In ACEX 1K and FLEX 10K devices, the initialization clock source is either an external host (e.g. a configuration device or microprocessor) or the optional CLKUSR pin. By default, an external host must provide the initialization clock on the DCLK pin. Programming files generated by the Quartus II or MAX+PLUS II software already have these initialization clock cycles included in the file.

You also have the flexibility to synchronize initialization of multiple devices by using the CLKUSR option. You can turn on the *Enable user-supplied start-up clock (CLKUSR)* option in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR will not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, Mercury devices require 136 clock cycles to initialize properly, APEX 20K (2.5 V) devices require 40 clock cycles, ACEX 1K and FLEX 10K devices require 10 clock cycles.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The *Enable INIT_DONE output* option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used, it will be high due to an external 1-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. This low-to-high transition signals that the FPGA has entered user mode. In user-mode, the user I/O pins will no longer have weak pull-up resistors and will function as assigned in your design. The enhanced configuration device and EPC2 device drive DCLK low and DATA high (EPC1 devices tri-state DATA) at the end of configuration.

If an error occurs during configuration, the FPGA drives its nSTATUS pin low, resetting itself internally. Since the nSTATUS pin is tied to OE, the configuration device will also be reset. If the *Auto-Restart Configuration After Error* option available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box is turned on, the FPGA automatically initiates reconfiguration if an error occurs. The Mercury, APEX 20K (2.5 V), ACEX 1K, or FLEX 10K device releases its nSTATUS pin after a reset time-out period (maximum of 40 µs). When the nSTATUS pin is released and pulled high by a pull-up resistor, the configuration device reconfigures the chain. If this option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low to restart configuration. The external system can pulse nCONFIG if nCONFIG is under system control rather than tied to V_{CC}.

In addition, if the configuration device sends all of its data and then detects that CONF_DONE has not gone high, it recognizes that the FPGA has not configured successfully. Enhanced configuration devices wait for 64 DCLK cycles after the last configuration bit was sent for CONF_DONE to reach a high state. EPC1 and EPC2 devices wait for 16 DCLK cycles. In this case, the configuration device pulls its OE pin low, which in turn drives the target device's nSTATUS pin low. If the *Auto-Restart Configuration After Error* option is set in the software, the target device resets and then releases its nSTATUS pin after a reset time-out period (maximum of 40 µs). When nSTATUS returns high, the configuration device tries to reconfigure the FPGA.

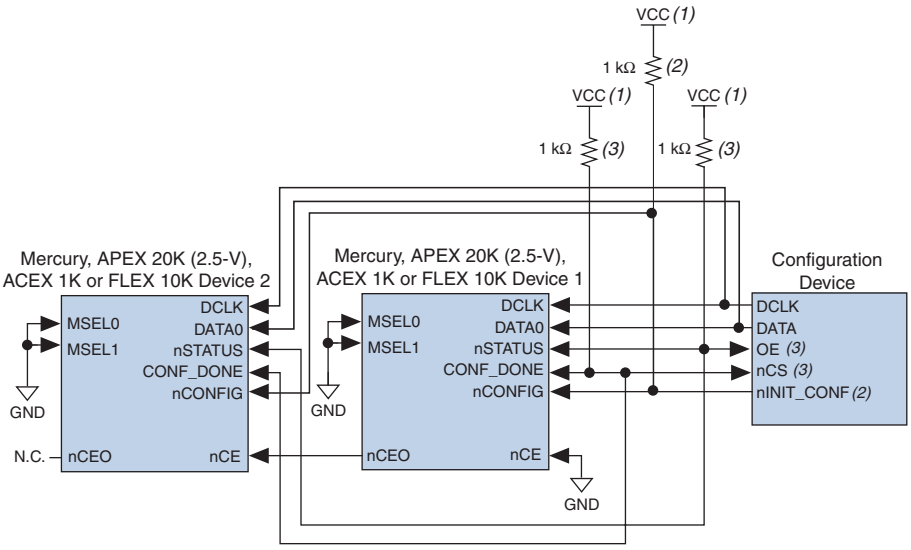
When CONF_DONE is sensed low after configuration, the configuration device recognizes that the target device has not configured successfully; therefore, your system should not pull CONF_DONE low to delay initialization. Instead, use the CLKUSR option to synchronize the initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain will initialize together if their CONF_DONE pins are tied together.

If the optional CLKUSR pin is being used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure that CLKUSR continues toggling during the time nSTATUS is low (maximum of 40 µs).

When the FPGA is in user-mode, a reconfiguration can be initiated by pulling the nCONFIG pin low. When nCONFIG is pulled low, the FPGA also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Since CONF_DONE is pulled low, this activates the configuration device since it will see its nCS pin drive low. Once nCONFIG returns to a logic high state and nSTATUS is released by the FPGA, reconfiguration begins.

Figure 8–2 shows how to configure multiple devices with a configuration device. This circuit is similar to the configuration device circuit for a single device, except Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices are cascaded for multi-device configuration.

Figure 8–2. Multi-Device PS Configuration Using a Configuration Device



Notes to Figure 8–2:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) has an internal pull-up resistor that is always active, meaning an external pull-up resistor is not required on the nINIT_CONF/nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the *Disable nCS and OE pull-ups on configuration device* option when generating programming files.

When performing multi-device configuration, you must generate the configuration device's Programmer Object File (**.pof**) from each project's SRAM Object File (**.sof**). You can combine multiple SOFs using the Quartus For more information on how to create configuration files for multi-device configuration chains, refer to *Software Settings*, chapter 6 and 7 in volume 2 of the *Configuration Handbook*.

In multi-device PS configuration, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they can require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DCLK and DATA lines are buffered for every fourth device.

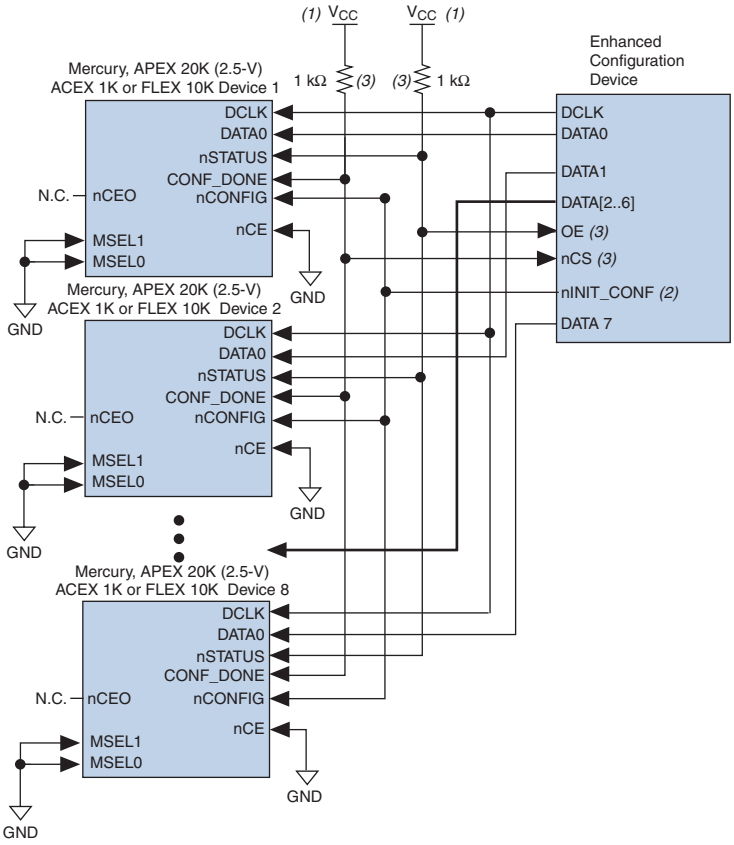
When configuring multiple devices, configuration does not begin until all devices release their OE or nSTATUS pins. Similarly, since all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Since all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first FPGA flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This low signal drives the OE pin low on the configuration device and drives nSTATUS low on all FPGAs, which causes them to enter a reset state. This behavior is similar to a single FPGA detecting an error.

If the *Auto-Restart Configuration After Error* option is turned on, the devices will automatically initiate reconfiguration if an error occurs. The FPGAs will release their nSTATUS pins after a reset time-out period (maximum of 40 µs). When all the nSTATUS pins are released and pulled high, the configuration device tries to reconfigure the chain. If the *Auto-Restart Configuration After Error* option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low to restart configuration. The external system can pulse nCONFIG if nCONFIG is under system control rather than tied to V_{CC}.

Enhanced configuration devices also support parallel configuration of up to eight devices. The n-bit (n = 1, 2, 4, or 8) PS configuration mode allows enhanced configuration devices to concurrently configure FPGAs or a chain of FPGAs. In addition, these devices do not have to be the same device family or density; they can be any combination of Altera FPGAs. An individual enhanced configuration device DATA line is available for each targeted FPGA. Each DATA line can also feed a daisy chain of FPGAs. Figure 8–3 shows how to concurrently configure multiple devices using an enhanced configuration device.

Figure 8–3. Concurrent PS Configuration of Multiple Devices Using an Enhanced Configuration Device



Notes to Figure 8–3:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) has an internal pull-up resistor that is always active, meaning an external pull-up resistor is not required on the nINIT_CONF/nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the *Disable nCS and OE pull-ups on configuration device* option when generating programming files.

The Quartus II software only allows the selection of n-bit PS configuration modes, where n must be 1, 2, 4, or 8. However, you can use these modes to configure any number of devices from 1 to 8. When configuring SRAM-based devices using n-bit PS modes, use Table 8–6 to select the appropriate configuration mode for the fastest configuration times.

Table 8–6. Recommended Configuration Using n-Bit PS Modes

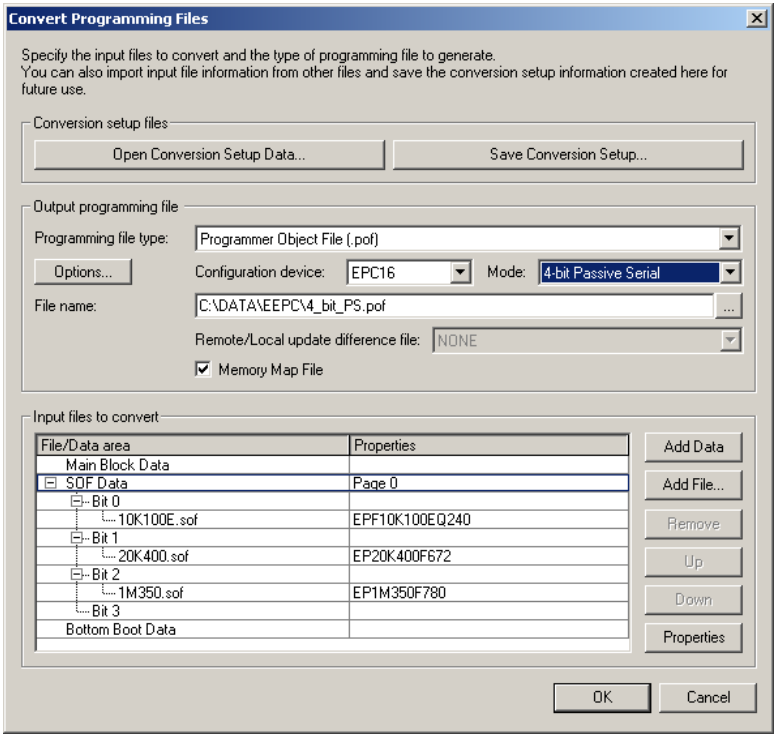
Number of Devices *(1)*	Recommended Configuration Mode
1	1-bit PS
2	2-bit PS
3	4-bit PS
4	4-bit PS
5	8-bit PS
6	8-bit PS
7	8-bit PS
8	8-bit PS

Note to Table 8–6:

(1) Assume that each DATA line is only configuring one device, not a daisy chain of devices.

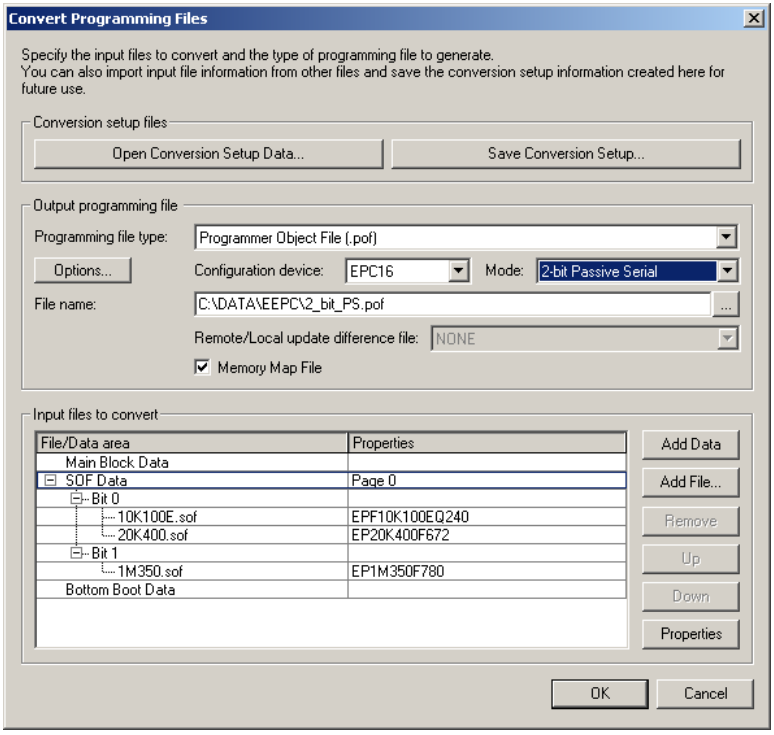
For example, if you configure three FPGAs, you would use the 4-bit PS mode. For the DATA0, DATA1, and DATA2 lines, the corresponding SOF data is transmitted from the configuration device to the FPGA. For DATA3, you can leave the corresponding Bit3 line blank in the Quartus II software. On the printed circuit board (PCB), leave the DATA3 line from the enhanced configuration device unconnected. Figure 8–4 shows the Quartus II **Convert Programming Files** window (Tools menu) setup for this scheme.

Figure 8–4. Software Settings for Configuring Devices Using n-Bit PS Modes



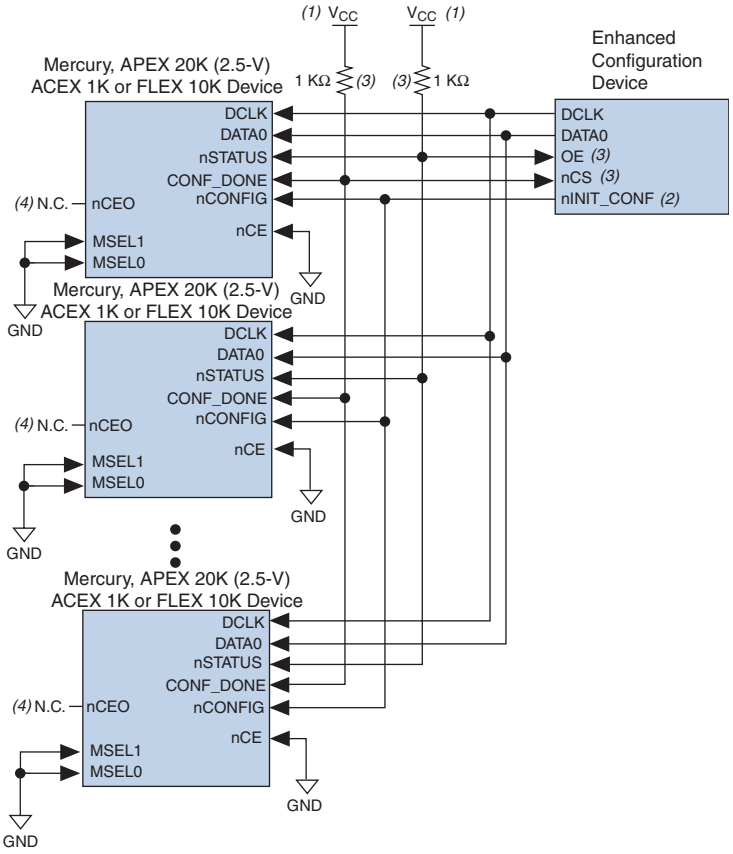
Alternatively, you can daisy chain two FPGAs to one DATA line while the other DATA lines drive one device each. For example, you could use the 2-bit PS mode to drive two FPGAs with DATA Bit0 (EPF10K100E and EP20K400 devices) and the third device (the EP1M350 device) with DATA Bit1. This 2-bit PS configuration scheme requires less space in the configuration flash memory, but can increase the total system configuration time (Figure 8–5).

Figure 8–5. Software Settings for Daisy Chaining Two FPGAs on One DATA Line



In your system, you may have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they can require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete configuration at the same time. Figure 8–6 shows multi-device PS configuration when the Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices are receiving the same configuration data.

Figure 8–6. Multiple-Device PS Configuration Using an Enhanced Configuration Device When FPGAs Receive the Same Data



Notes to Figure 8–6:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) has an internal pull-up resistor that is always active, meaning an external pull-up resistor is not required on the nINIT_CONF/nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the *Disable nCS and OE pull-ups on configuration device* option when generating programming files.

(4) The nCEO pins of all devices are left unconnected when configuring the same configuration data into multiple devices.

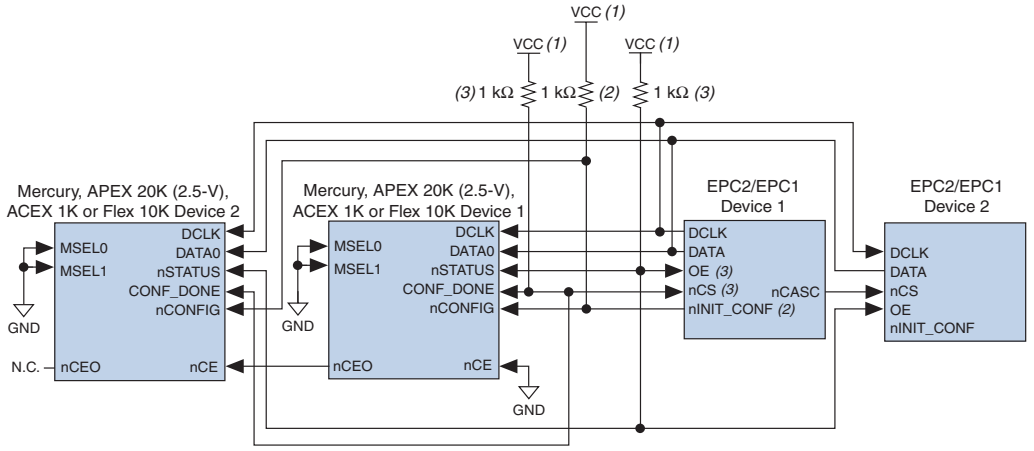
You can cascade several EPC2 or EPC1 devices to configure multiple Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices. The first configuration device in the chain is the master configuration device, while the subsequent devices are the slave devices. The master configuration device sends DCLK to the Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices and to the slave configuration devices. The first EPC device's nCS pin is connected to the CONF_DONE pins of the FPGAs, while its nCASC pin is connected to nCS of the next configuration device in the chain. The last device's nCS input comes from the previous device, while its nCASC pin is left floating. When all data from the first configuration device is sent, it drives nCASC low, which in turn drives nCS on the next configuration device. Because a configuration device requires less than one clock cycle to activate a subsequent configuration device, the data stream is uninterrupted.

Enhanced configuration devices EPC4, EPC8, and EPC16 cannot be cascaded.

Since all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, the master configuration device stops configuration for the entire chain and the entire chain must be reconfigured. For example, if the master configuration device does not detect CONF_DONE going high at the end of configuration, it resets the entire chain by pulling its OE pin low. This low signal drives the OE pin low on the slave configuration device(s) and drives nSTATUS low on all FPGAs, causing them to enter a reset state. This behavior is similar to the FPGA detecting an error in the configuration data.

Figure 8–7 shows how to configure multiple devices using cascaded EPC2 or EPC1 devices.

Figure 8–7. Multi-Device PS Configuration Using Cascaded EPC2 or EPC1 Devices



Notes to Figure 8–7:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) has an internal pull-up resistor that is always active, meaning an external pull-up resistor is not required on the nINIT_CONF/nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the *Disable nCS and OE pull-ups on configuration device* option when generating programming files.

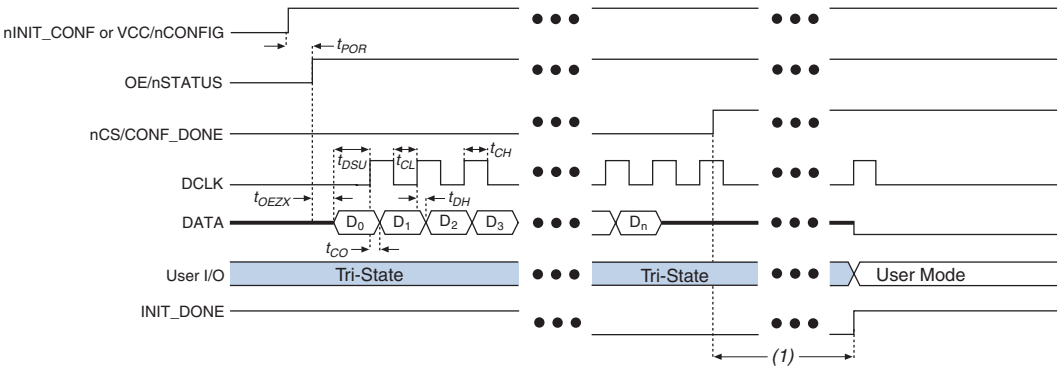
When using enhanced configuration devices or EPC2 devices, nCONFIG of the FPGA can be connected to nINIT_CONF, which allows the INIT_CONF JTAG instruction to initiate FPGA configuration. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CC} either directly or through a resistor. An internal pull-up on the nINIT_CONF pin is always active in the enhanced configuration devices and the EPC2 devices, which means an external pull-up is not required if nCONFIG is tied to nINIT_CONF. If multiple EPC2 devices are used to configure a Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K device(s), only the first EPC2 has its nINIT_CONF pin tied to the device's nCONFIG pin.

You can use a single configuration chain to configure Mercury, APEX 20K, ACEX 1K, and FLEX 10KE devices with other Altera devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information on configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains*, chapter 8 in volume 2 of the *Configuration Handbook*.

Figure 8–8 shows the timing waveform for the PS configuration scheme using a configuration device.

Figure 8–8. PS Configuration Using a Configuration Device Timing Waveform



Note to Figure 8–8:

(1) Mercury devices enter user-mode 136 clock cycles after CONF_DONE goes high. APEX 20K devices enter user-mode 40 clock cycles after CONF_DONE goes high. The initialization clock can come from the Mercury or APEX 20K internal oscillator or the CLKUSR pin. ACEX 1K and FLEX 10K devices enter user-mode 10 clock cycles after CONF_DONE goes high. The initialization clock can come from DCLK or CLKUSR.

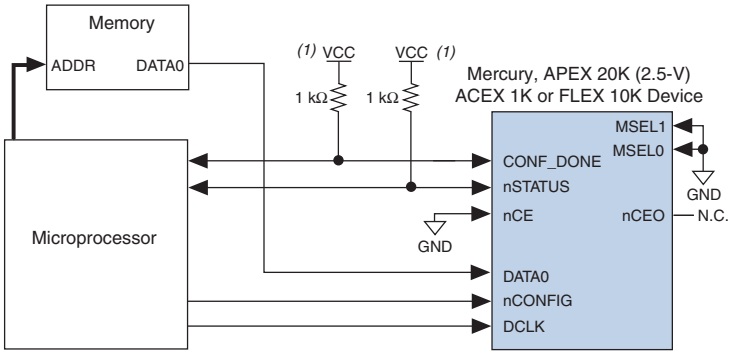
For timing information, refer to the *Altera Configuration Devices*, chapter 1 the *Enhanced Configuration Devices (EPC4, EPC8, and EPC16) Data Sheet*, chapter 2, or the *Configuration Devices for SRAM-based LUT Devices Data Sheet* chapter 5 in volume 2 of the *Configuration Handbook*.

Device configuration options and how to create configuration files are discussed further in *Software Settings*, chapters 6 and 7 in volume 2 of the *Configuration Handbook*.

PS Configuration Using a Microprocessor

In the PS configuration scheme, an intelligent host (e.g., a microprocessor or CPLD) can transfer configuration data from a storage device (e.g., flash memory) to the target Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices. Configuration data can be stored in RBF, HEX, or TTF format. Figure 8–9 shows the configuration interface connections between the Mercury, APEX 20K (2.5 V), ACEX 1K, or FLEX 10K device and a microprocessor for single device configuration.

Figure 8–9. Single Device PS Configuration Using a Microprocessor



Note to Figure 8–9:
(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for the device.

Upon power-up, the Mercury, APEX 20K (2.5 V), ACEX 1K, or FLEX 10K device goes through a POR for approximately 5 µs. During POR, the device resets and holds nSTATUS low, and tri-states all user I/O pins. Once the FPGA successfully exits POR, all user I/O pins are tri-stated. Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10KE devices have weak pull-up resistors on the user I/O pins which are on before and during configuration.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the Operating Conditions table of the appropriate device family data sheet.

The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration, the microprocessor must generate a low-to-high transition on the nCONFIG pin.

VCCINT and VCCIO pins on the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 1-kΩ pull-up resistor. Once nSTATUS is released, the FPGA is ready to receive configuration data and the configuration stage begins. When nSTATUS is pulled high, the microprocessor should place the configuration data one bit at a time on the DATA0 pin. The least significant bit (LSB) of each data byte must be sent first.

The Mercury, APEX 20K (2.5 V), ACEX 1K, or FLEX 10K device receives configuration data on its DATA0 pin and the clock is received on the DCLK pin. Data is latched into the FPGA on the rising edge of DCLK. Data is continuously clocked into the target device until CONF_DONE goes high. After the FPGA has received all configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by an external 1-kΩ pull-up resistor. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin.

In Mercury and APEX 20K (2.5 V) devices, the initialization clock source is either the FPGA's internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Mercury or APEX 20K (2.5 V) device allows enough clock cycles for proper initialization.

In ACEX 1K and FLEX 10K devices, the initialization clock source is either an external host (e.g. a configuration device or microprocessor) or the optional CLKUSR pin. By default, the clock on DCLK is the clock source for initialization. Programming files generated by the Quartus II or MAX+PLUS II software already have these initialization clock cycles included in the file. Therefore, sending the entire configuration file to the device is sufficient to configure and initialize the device. Driving DCLK to the device after configuration is complete does not affect device operation.

You also have the flexibility to synchronize initialization of multiple devices by using the CLKUSR option. The *Enable user-supplied start-up clock (CLKUSR)* option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR does not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, Mercury devices require 136 clock cycles to initialize properly, APEX 20K (2.5 V) devices require 40 clock cycles, ACEX 1K and FLEX 10K devices require 10 clock cycles.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The *Enable INIT_DONE output* option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used it will be high due to an external 1-kΩ pull-up when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low. When initialization is complete, the INIT_DONE pin will be released and pulled high. The microprocessor must be able to detect this low-to-high transition which signals the FPGA has entered user mode. In user-mode, the user I/O pins will no longer have weak pull-up resistors and will function as assigned in your design. To ensure DCLK and DATA are not left floating at the end of configuration, the microprocessor must drive them either high or low, whichever is convenient on your board.

Handshaking signals are not used in PS configuration mode. Therefore, the configuration clock (DCLK) speed must be below the specified frequency to ensure correct configuration. No maximum DCLK period exists, which means you can pause configuration by halting DCLK for an indefinite amount of time.

If an error occurs during configuration, the FPGA drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the microprocessor that there is an error. If the *Auto-Restart Configuration After Error* option (available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box) is turned on, the Mercury, APEX 20K (2.5 V), ACEX 1K, or FLEX 10K device releases nSTATUS after a reset time-out period (maximum of 40 µs). After nSTATUS is released and pulled high by a pull-up resistor, the microprocessor can try to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition on nCONFIG to restart the configuration process.

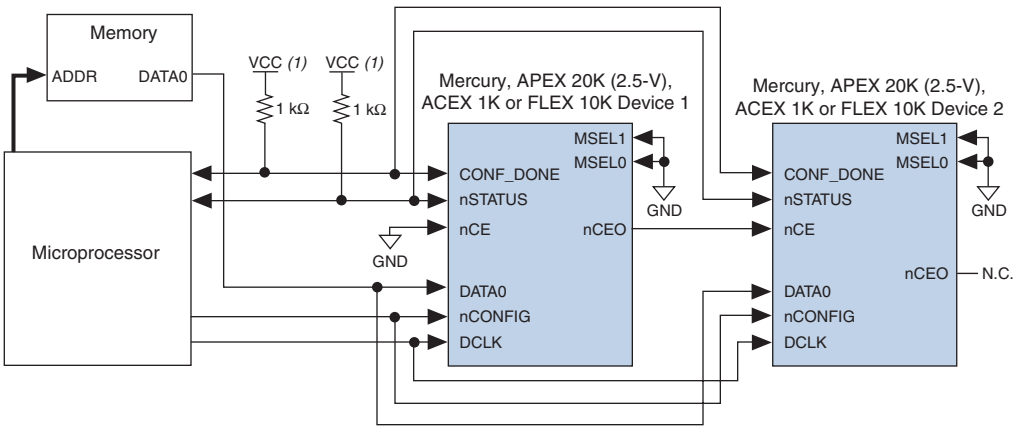
The microprocessor can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The CONF_DONE pin must be monitored by the microprocessor to detect errors and determine when programming completes. If the microprocessor sends all configuration data but CONF_DONE or INIT_DONE have not gone high, the microprocessor must reconfigure the target device.

If the optional CLKUSR pin is being used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure that CLKUSR continues toggling during the time nSTATUS is low (maximum of 40 µs).

When the FPGA is in user-mode, you can initiate a reconfiguration by transitioning the nCONFIG pin low-to-high. The nCONFIG pin should be low for at least 21 µs for Mercury devices, 8 µs for APEX 20K devices and 2 µs for ACEX 1K and Flex 10K devices. When nCONFIG is pulled low, the FPGA also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Once nCONFIG returns to a logic high state and nSTATUS is released by the FPGA, reconfiguration begins.

Figure 8–10 shows how to configure multiple devices using a microprocessor. This circuit is similar to the PS configuration circuit for a single device, except Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices are cascaded for multi-device configuration.

Figure 8–10. Multi-Device PS Configuration Using a Microprocessor



Note to Figure 8–10:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.

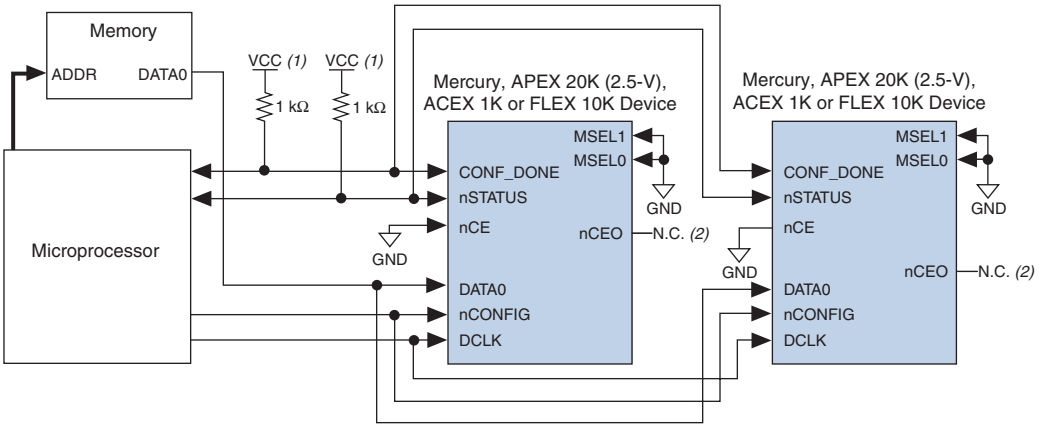
In multi-device PS configuration the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration within one clock cycle; therefore, the transfer of data destinations is transparent to the microprocessor. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they can require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DCLK and DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Since all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first FPGA flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single FPGA detecting an error.

If the *Auto-Restart Configuration After Error* option is turned on, the FPGAs release their nSTATUS pins after a reset time-out period (maximum of 40 µs). After all nSTATUS pins are released and pulled high, the microprocessor can try to reconfigure the chain without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition on nCONFIG to restart the configuration process.

In your system, you can have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they can require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete configuration at the same time. Figure 8–11 shows multi-device PS configuration when Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices are receiving the same configuration data.

Figure 8–11. Multiple-Device PS Configuration Using a Microprocessor When Both FPGAs Receive the Same Data



Notes to Figure 8–11:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.

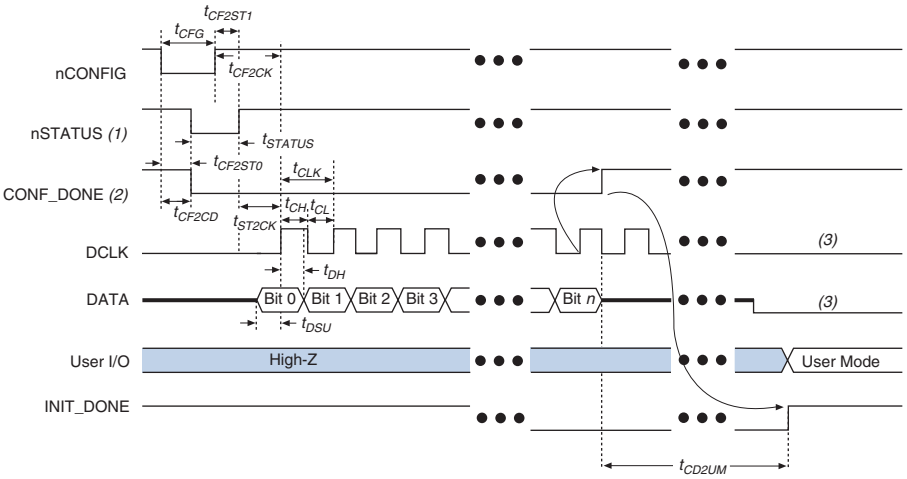
(2) The nCEO pins of both devices are left unconnected when configuring the same configuration data into multiple devices.

You can use a single configuration chain to configure Mercury, APEX 20K, ACEX 1K, and FLEX 10KE devices with other Altera devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information on configuring multiple Altera devices in the same configuration chain, refer to *Configuring Mixed Altera FPGA Chains* chapter 8 in volume 2 of the *Configuration Handbook*.

Figure 8–12 shows the timing waveform for the PS configuration for Mercury, APEX 20K, ACEX 1K, and FLEX 10KE devices when using a microprocessor.

Figure 8–12. PS Configuration Using a Microprocessor Timing Waveform



Notes to Figure 8–12:

(1) Upon power-up, the Mercury, APEX 20K, ACEX 1K, or FLEX 10K device holds nSTATUS low for not more than 5 µs after V_{CC} reaches its minimum requirement.

(2) Upon power-up, before and during configuration, CONF_DONE is low.

(3) DATA0 and DCLK should not be left floating after configuration. It should be driven high or low, whichever is more convenient.

Tables 8–7 through 8–10 defines the timing parameters for Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices for PS configuration.

Table 8–7. PS Timing Parameters for Mercury Devices (Part 1 of 2) *Note (1)*

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		200	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		200	ns
t_{CFG}	nCONFIG low pulse width	21		µs
t_{STATUS}	nSTATUS low pulse width	10	40 *(2)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high		1 *(2)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	45		µs
t_{ST2CK}	nSTATUS high to first rising edge on DCLK	1		µs
t_{DSU}	Data setup time before rising edge on DCLK	10		ns
t_{DH}	Data hold time after rising edge on DCLK	0		ns

Table 8–7. PS Timing Parameters for Mercury Devices (Part 2 of 2) *Note (1)*

Symbol	Parameter	Min	Max	Units
t_{CH}	DCLK high time	10		ns
t_{CL}	DCLK low time	10		ns
t_{CLK}	DCLK period	20		ns
f_{MAX}	DCLK frequency		50	MHz
t_{CD2UM}	CONF_DONE high to user mode *(3)*	6	28	µs

Notes to Table 8–7:
(1) This information is preliminary.
(2) This value is obtainable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.
(3) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device. If the clock source is CLKUSR, multiply the clock period by 136 to obtain this value.

Table 8–8. PS Timing Parameters for APEX 20K Devices

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		200	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		200	ns
t_{CFG}	nCONFIG low pulse width	8		µs
t_{STATUS}	nSTATUS low pulse width	10	40 *(1)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high		1 *(1)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	40		µs
t_{ST2CK}	nSTATUS high to first rising edge on DCLK	1		µs
t_{DSU}	Data setup time before rising edge on DCLK	10		ns
t_{DH}	Data hold time after rising edge on DCLK	0		ns
t_{CH}	DCLK high time	15		ns
t_{CL}	DCLK low time	15		ns
t_{CLK}	DCLK period	30		ns
f_{MAX}	DCLK maximum frequency		33.3	MHz
t_{CD2UM}	CONF_DONE high to user mode *(2)*	2	8	µs

Notes to Table 8–8:
(1) This value is obtainable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.
(2) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device. If the clock source is CLKUSR, multiply the clock period by 40 to obtain this value.

Table 8–9. PS Timing Parameters for ACEX & FLEX 10KE Devices

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		200	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		200	ns
t_{CFG}	nCONFIG low pulse width	2		µs
t_{STATUS}	nSTATUS low pulse width	1	10 *(1)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high		4 *(1)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	5		µs
t_{ST2CK}	nSTATUS high to first rising edge on DCLK	1		µs
t_{DSU}	Data setup time before rising edge on DCLK	10		ns
t_{DH}	Data hold time after rising edge on DCLK	0		ns
t_{CH}	DCLK high time	15		ns
t_{CL}	DCLK low time	15		ns
t_{CLK}	DCLK period	30		ns
f_{MAX}	DCLK maximum frequency		33.3	MHz
t_{CD2UM}	CONF_DONE high to user mode *(2)*	0.6	2	µs

Notes to Table 8–9:

(1) This value is obtainable if users do not delay configuration by extending the nSTATUS low pulse width.

(2) The minimum and maximum numbers apply only if DCLK is chosen as the clock source for starting up the device. If the clock source is CLKUSR, multiply the clock period by 10 to obtain this value.

Table 8–10. PS Timing Parameters for FLEX 10K Devices (Part 1 of 2)

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		200	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		200	ns
t_{CFG}	nCONFIG low pulse width	2		µs
t_{STATUS}	nSTATUS low pulse width	1	10 *(1)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high		4 *(1)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	5		µs
t_{ST2CK}	nSTATUS high to first rising edge on DCLK	1		µs
t_{DSU}	Data setup time before rising edge on DCLK	10		ns
t_{DH}	Data hold time after rising edge on DCLK	0		ns
t_{CH}	DCLK high time	30		ns
t_{CL}	DCLK low time	30		ns
t_{CLK}	DCLK period	60		ns

Table 8–10. PS Timing Parameters for FLEX 10K Devices (Part 2 of 2)

Symbol	Parameter	Min	Max	Units
f_{MAX}	DCLK maximum frequency		16.7	MHz
t_{CD2UM}	CONF_DONE high to user mode *(2)*	0.6	2	µs

Notes to Table 8–10:

(1) This value is obtainable if users do not delay configuration by extending the nSTATUS low pulse width.

(2) The minimum and maximum numbers apply only if DCLK is chosen as the clock source for starting up the device. If the clock source is CLKUSR, multiply the clock period by 10 to obtain this value.

Device configuration options and how to create configuration files are discussed further in *Software Settings*, chapters 6 and 7 in Volume 2 of the *Configuration Handbook*.

Configuring Using the MicroBlaster Driver

The MicroBlaster™ software driver allows you to configure Altera's FPGAs through the ByteBlasterMV cable in PS mode. The MicroBlaster software driver supports a RBF programming input file and is targeted for embedded passive serial configuration. The source code is developed for the Windows NT operating system, although you can customize it to run on other operating systems. For more information on the MicroBlaster software driver, go to the Altera web site (**http://www.altera.com**).

PS Configuration Using a Download Cable

In this section, the generic term "download cable" includes the Altera USB Blaster universal serial bus (USB) port download cable, MasterBlaster™ serial/USB communications cable, ByteBlaster™ II parallel port download cable, and the ByteBlasterMV™ parallel port download cable.

In PS configuration with a download cable, an intelligent host (e.g., a PC) transfers data from a storage device to the FPGA via the USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV cable.

Upon power-up, the Mercury, APEX 20K (2.5 V), ACEX 1K, or FLEX 10K device goes through a POR for approximately 5 µs. During POR, the device resets and holds nSTATUS low, and tri-states all user I/O pins. Once the FPGA successfully exits POR, all user I/O pins are tri-stated. Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10KE devices have weak pull-up resistors on the user I/O pins which are on before and during configuration.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the Operating Conditions table of the appropriate device family data sheet.

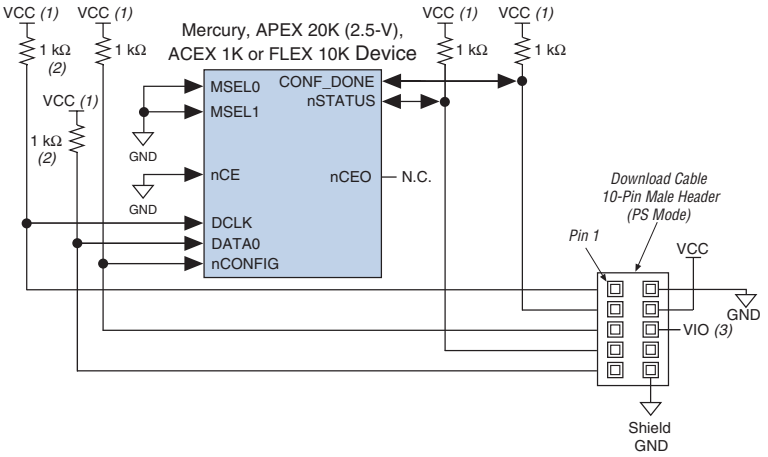
The configuration cycle consists of 3 stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration in this scheme, the download cable generates a low-to-high transition on the nCONFIG pin.

VCCINT and VCCIO pins on the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 1-kΩ pull-up resistor. Once nSTATUS is released the FPGA is ready to receive configuration data and the configuration stage begins. The programming hardware or download cable then places the configuration data one bit at a time on the device's DATA0 pin. The configuration data is clocked into the target device until CONF_DONE goes high.

When using a download cable, setting the *Auto-Restart Configuration After Error* option does not affect the configuration cycle because you must manually restart configuration in the Quartus II software when an error occurs. Additionally, the *Enable user-supplied start-up clock (CLKUSR)* option has no affect on the device initialization since this option is disabled in the SOF when programming the FPGA using the Quartus II programmer and download cable. Therefore, if you turn on the *CLKUSR* option, you do not need to provide a clock on CLKUSR when you are configuring the FPGA with the Quartus II programmer and a download cable. Figure 8–13 shows PS configuration for Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices using a USB Blaster, MasterBlaster, ByteBlaster II or ByteBlasterMV cable.

Figure 8–13. PS Configuration Using a USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV Cable



Notes to Figure 8–13:

(1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster (VIO pin), ByteBlaster II or ByteBlasterMV cable.

(2) The pull-up resistors on DATA0 and DCLK are only needed if the download cable is the only configuration scheme used on your board. This is to ensure that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, the pull-up resistors on DATA0 and DCLK are not needed.

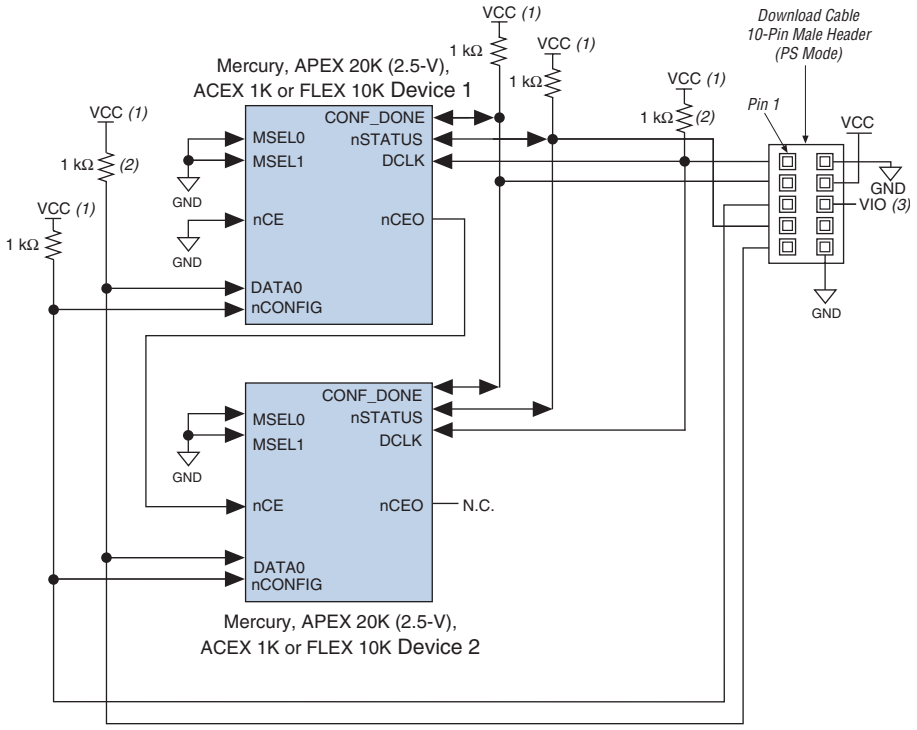
(3) Pin 6 of the header is a VIO reference voltage for the MasterBlaster output driver. VIO should match the device's V_{CCIO}. Refer to the MasterBlaster Serial/USB Communications Cable Data Sheet for this value. In the ByteBlasterMV, this pin is a no connect. In the USB Blaster and ByteBlaster II, this pin is connected to nCE when it is used for Active Serial programming, otherwise it is a no connect.

You can use a download cable to configure multiple Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices by connecting each device's nCEO pin to the subsequent device's nCE pin. The first device's nCE pin is connected to GND while its nCEO pin is connected to the nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. All other configuration pins, nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE are connected to every device in the chain. Because all CONF_DONE pins are tied together, all devices in the chain initialize and enter user mode at the same time.

In addition, because the nSTATUS pins are tied together, the entire chain halts configuration if any device detects an error. The *Auto-Restart Configuration After Error* option does not affect the configuration cycle because you must manually restart configuration in the Quartus II software when an error occurs.

Figure 8–14 shows how to configure multiple Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices with a download cable.

Figure 8–14. Multi-Device PS Configuration Using a USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV Cable



Notes to Figure 8–14:

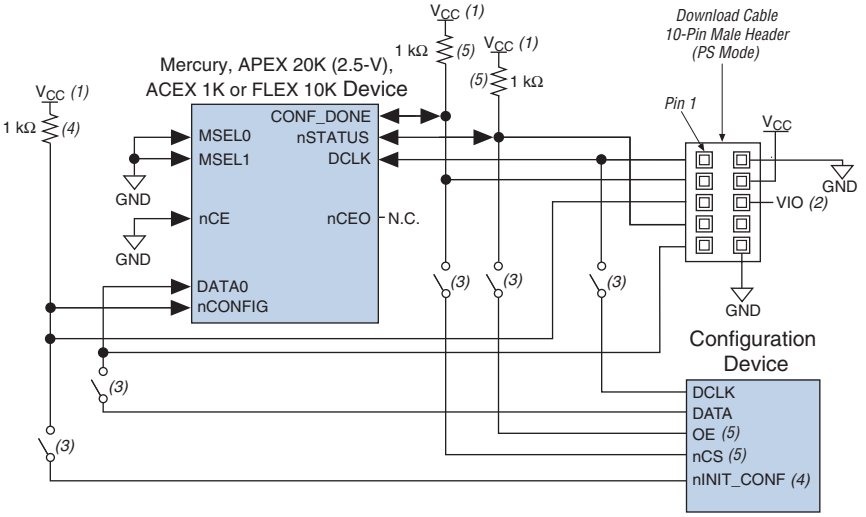
(1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster (VIO pin), ByteBlaster II or ByteBlasterMV cable.

(2) The pull-up resistors on DATA0 and DCLK are only needed if the download cable is the only configuration scheme used on your board. This is to ensure that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, the pull-up resistors on DATA0 and DCLK are not needed.

(3) Pin 6 of the header is a VIO reference voltage for the MasterBlaster output driver. VIO should match the device's V_{CCIO}. Refer to the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value. In the ByteBlasterMV, this pin is a no connect. In the USB Blaster and ByteBlaster II, this pin is connected to nCE when it is used for Active Serial programming, otherwise it is a no connect.

If you are using a download cable to configure device(s) on a board that also has configuration devices, you should electrically isolate the configuration device from the target device(s) and cable. One way to isolate the configuration device is to add logic, such as a multiplexer, that can select between the configuration device and the cable. The multiplexer chip should allow bidirectional transfers on the nSTATUS and CONF_DONE signals. Another option is to add switches to the five common signals (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) between the cable and the configuration device. The last option is to remove the configuration device from the board when configuring the FPGA with the cable. Figure 8–15 shows a combination of a configuration device and a download cable to configure an FPGA.

Figure 8–15. PS Configuration with a Download Cable & Configuration Device Circuit



Notes to Figure 8–15:

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) Pin 6 of the header is a VIO reference voltage for the MasterBlaster output driver. VIO should match the device's V_{CCIO}. Refer to the MasterBlaster Serial/USB Communications Cable Data Sheet for this value. In the ByteBlasterMV, this pin is a no connect. In the USB Blaster and ByteBlaster II, this pin is connected to nCE when it is used for Active Serial programming, otherwise it is a no connect.

(3) You should not attempt configuration with a download cable while a configuration device is connected to a Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K device. Instead, you should either remove the configuration device from its socket when using the download cable or place a switch on the five common signals between the download cable and the configuration device.

(4) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) has an internal pull-up resistor that is always active. This means an external pull-up resistor is not required on the nINIT_CONF/nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available (e.g., on EPC1 devices), nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(5) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the *Disable nCS and OE pull-up resistors on configuration device* option when generating programming files.

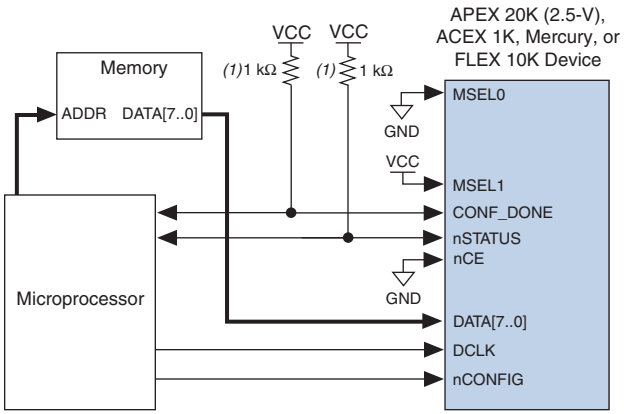
For more information on how to use the USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV cables, refer to the following data sheets:

- *USB Blaster USB Port Download Cable Data Sheet*
- *MasterBlaster Serial/USB Communications Cable Data Sheet*
- *ByteBlaster II Parallel Port Download Cable Data Sheet*
- *ByteBlasterMV Parallel Port Download Cable Data Sheet*

Passive Parallel Synchronous Configuration

Passive parallel synchronous (PPS) configuration uses an intelligent host, such as a microprocessor, to transfer configuration data from a storage device, such as flash memory, to the target Mercury, APEX 20K (2.5 V), ACEX 1K, or FLEX 10K device. Configuration data can be stored in TTF, RBF, or HEX format. The host system outputs byte-wide data and the serializing clock to the FPGA. The target device latches the byte-wide data on the DATA[7..0] pins on the rising edge of DCLK and then uses the next eight falling edges on DCLK to serialize the data internally. On the ninth rising DCLK edge, the next byte of configuration data is latched into the target device. Figure 8–16 shows the configuration interface connections between the FPGA and a microprocessor for single device configuration.

Figure 8–16. Single Device PPS Configuration Using a Microprocessor



Note to Figure 8–16:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for the device.

Upon power-up, the Mercury, APEX 20K (2.5 V), ACEX 1K or FLEX 10K device goes through a Power-On Reset (POR) for approximately 5 µs. During POR, the device resets and holds nSTATUS low, and tri-states all user I/O pins. Once the FPGA successfully exits POR, all user I/O pins are tri-stated. Mercury, APEX 0K (2.5 V), ACEX 1K, and FLEX 10K devices have weak pull-up resistors on the user I/O pins which are on before and during configuration.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the Operating Conditions table of the appropriate device family data sheet.

The configuration cycle consists of 3 stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration in this scheme, the microprocessor must generate a low-to-high transition on the nCONFIG pin.

VCCINT and VCCIO pins on the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 1-kΩ pull-up resistor. Once nSTATUS is released the FPGA is ready to receive configuration data and the configuration stage begins. When nSTATUS is pulled high, the microprocessor should place the configuration data one byte at a time on the DATA[7..0] pins. New configuration data should be sent to the FPGA every eight DCLK cycles.

The Mercury, APEX 20K (2.5 V), ACEX 1K or FLEX 10K device receives configuration data on its DATA[7..0] pins and the clock is received on the DCLK pin. On the first rising DCLK edge, a byte of configuration data is latched into the target device; the subsequent eight falling DCLK edges serialize the configuration data in the device. On the ninth rising clock edge, the next byte of configuration data is latched and serialized into the target device.

Data is clocked into the target device until CONF_DONE goes high. After the FPGA has received all configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by an external 1-kΩ pull-up resistor. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin.

In Mercury and APEX 20K devices, the initialization process is synchronous and can be clocked by its internal oscillator (typically 10 MHz) or by the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Mercury or APEX 20K device will take care to provide itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device. Driving DCLK to the device after configuration is complete does not affect device operation.

In ACEX 1K and FLEX 10K devices, initialization can be clocked by the clock input on the DCLK pin or by the optional CLKUSR pin. By default, the clock on DCLK is the clock source for initialization. The configuration files created by the Quartus II and the MAX+PLUS II software incorporate the extra bits for proper device initialization. Therefore, sending the entire configuration file to the device is sufficient to configure and initialize the device. Driving DCLK to the device after configuration is complete does not affect device operation.

You also have the flexibility to synchronize initialization of multiple devices by using the CLKUSR option. The *Enable user-supplied start-up clock (CLKUSR)* option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR will not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, Mercury devices require 136 clock cycles to initialize properly. APEX 20K devices require 40 clock cycles, while ACEX 1K and FLEX 10K devices require 10 clock cycles.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. This *Enable INIT_DONE output* option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used it will be high due to an external 1-kΩ pull-up when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low. When initialization is complete, the INIT_DONE pin will be released and pulled high. The microprocessor must be able to detect this low-to-high transition which signals the FPGA has entered user mode. In user-mode, the user I/O pins will no longer have weak pull-ups and will function as assigned in your design. When initialization is complete, the FPGA enters user mode.

To ensure DCLK and DATA0 are not left floating at the end of configuration, the microprocessor must take care to drive them either high or low, whichever is convenient on your board. The DATA[7..1] pins are available as user I/O pins after configuration. When the PPS scheme is chosen in the Quartus II software, as a default these I/O pins are tri-stated in user mode and should be driven by the microprocessor. To change this default option in the Quartus II software, select the **Dual-Purpose Pins** tab of the **Device & Pin Options** dialog box.

The configuration clock (DCLK) speed must be below the specified frequency, as listed in Tables 8–12 through 8–14, to ensure correct configuration. No maximum DCLK period exists, which means you can pause configuration by halting DCLK for an indefinite amount of time. An optional status pin (RDYnBSY) on the FPGA indicates when it is busy serializing configuration data and when it is ready to accept the next data byte. The RDYnBSY pin is not required in the PPS mode. Configuration data can be sent every 8 DCLK cycles without monitoring this status pin.

If an error occurs during configuration, the FPGA drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the microprocessor that there is an error. If the *Auto-Restart Configuration on Error* option-available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box-is turned on, the FPGA releases nSTATUS after a reset time-out period (maximum of 40 µs). After nSTATUS is released and pulled high by a pull-up resistor, the microprocessor can try to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition on nCONFIG to restart the configuration process.

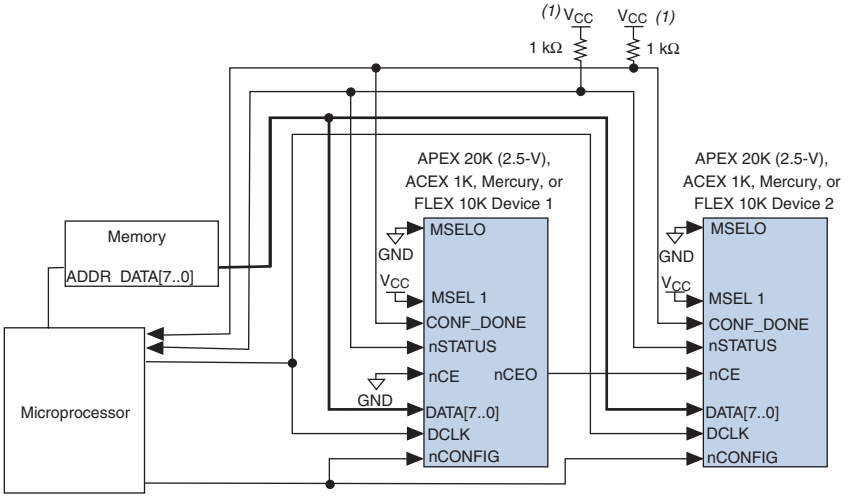
The microprocessor can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The CONF_DONE pin must be monitored by the microprocessor to detect errors and determine when programming completes. If the microprocessor sends all configuration data but CONF_DONE or INIT_DONE have not gone high, the microprocessor must reconfigure the target device.

If the optional CLKUSR pin is being used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure CLKUSR continues toggling during the time nSTATUS is low (maximum of 40 µs).

When the FPGA is in user-mode, a reconfiguration can be initiated by transitioning the nCONFIG pin low-to-high. The nCONFIG pin should be low for at least 21 µs for Mercury devices, 8 µs for APEX 20K devices and 2 µs for ACEX 1K and FLEX 10K devices. When nCONFIG is pulled low, the FPGA also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Once nCONFIG returns to a logic high state and nSTATUS is released by the FPGA, reconfiguration begins.

Figure 8–17 shows how to configure multiple Mercury, APEX 20K (2.5 V), ACEX 1K or FLEX 10K devices using a microprocessor. This circuit is similar to the PPS configuration circuit for a single device, except the devices are cascaded for multi-device configuration.

Figure 8–17. Multi-Device PPS Configuration Using a Microprocessor



Note to Figure 8–17:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.

In multi-device PPS configuration the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration within one clock cycle; therefore, the transfer of data destinations is transparent to the microprocessor.

Altera recommends keeping the configuration data valid on the DATA[7..0] bus for the 8 serializing clock cycles. The configuration data should be held valid on the DATA bus for the complete byte period because the nCEO of the first (and preceding) device can go low during the serializing DCLK cycles. Once the nCEO of the first (and preceding) device goes low, the second (and next) device becomes active and will begin trying to accept configuration data. If the configuration data is not valid on the first DCLK edge after nCEO goes low, then the second device will see incorrect configuration data and will never begin accepting configuration data. This situation will only arise if you are sharing the DATA[7..0] bus with other system data such that the configuration data is only valid for a portion of the byte period.

If your system requires to bus-share the DATA[7..0] line, you can work-around this by ensuring that the second (or next) device sees correct configuration data on the first rising edge of DCLK after the nCEO signal goes low. This can be achieved by delaying the nCEO signal by using external registers or by presenting the next byte of configuration data after the nCEO transition.

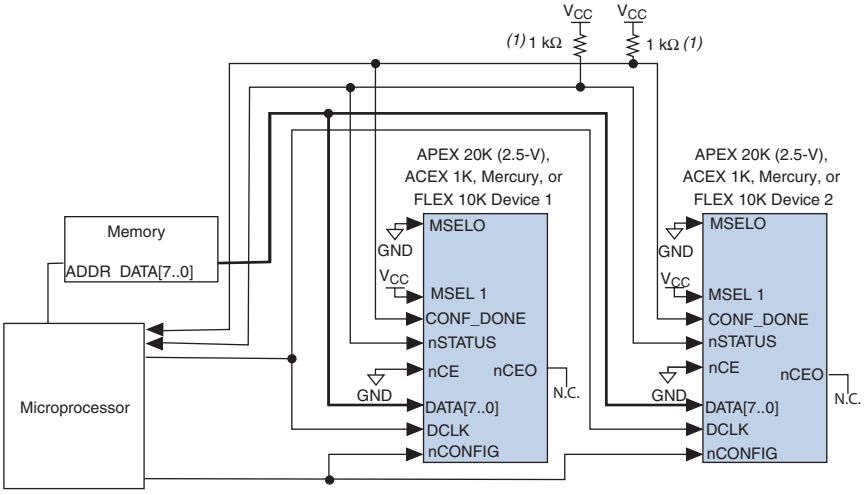
All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they may require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DCLK and DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Since all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first FPGA flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single FPGA detecting an error.

If the *Auto-Restart Configuration on Error* option is turned on, the FPGAs release their nSTATUS pins after a reset time-out period (maximum of 40 µs). After all nSTATUS pins are released and pulled high, the microprocessor can try to reconfigure the chain without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition on nCONFIG to restart the configuration process.

In your system, you may have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they may require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete configuration at the same time. Figure 8–18 shows multi-device PPS configuration when both devices are receiving the same configuration data.

Figure 8–18. Multiple-Device PPS Configuration Using a Microprocessor When Both FPGAs Receive the Same Data



Notes to Figure 8–18:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.

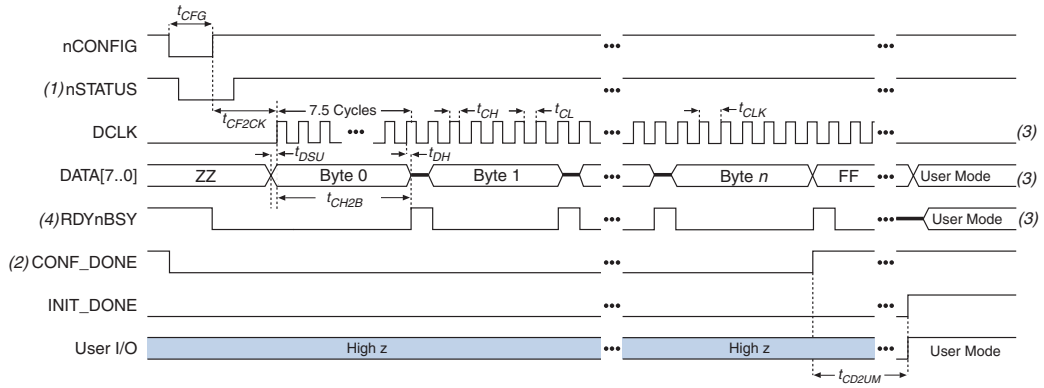
(2) The nCEO pins of both devices are left unconnected when configuring the same configuration data into multiple devices.

You can use a single configuration chain to configure Mercury, APEX 20K (2.5 V), ACEX 1K or FLEX 10KE devices with other Altera devices that support PPS configuration, such as APEX 20KE devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information on configuring multiple Altera devices in the same configuration chain, refer to *Configuring Mixed Altera FPGA Chains* chapter 8 in volume 2 of the *Configuration Handbook*.

Figure 8–19 shows the timing waveform for the PPS configuration scheme using a microprocessor.

Figure 8–19. Mercury, APEX 20K (2.5 V), ACEX 1K & FLEX 10KE PPS Configuration Timing Waveform



Notes to Figure 8–19:

(1) Upon power-up, the Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10KE device holds nSTATUS low for approximately 5 µs after V_{CC} reaches its minimum requirement.

(2) Upon power-up, before and during configuration, CONF_DONE is low.

(3) DATA0 and DCLK should not be left floating after configuration. It should be driven high or low, whichever is more convenient. DATA[7..1] and RDYnBSY are available as user I/O pins after configuration and the state of theses pins depends on the design programmed into the device.

(4) The RDYnBSY pin is not required in the PPS mode. Configuration data can be sent every 8 DCLK cycles without monitoring this status pin.

Tables 8–12 through 8–14 define the timing parameters for Mercury, APEX 20K, ACEX 1K, and FLEX 10K devices for PPS configuration.

Table 8–11. PPS Timing Parameters for Mercury Devices (Part 1 of 2)

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		200	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		200	ns
t_{CFG}	nCONFIG low pulse width	21		µs
t_{STATUS}	nSTATUS low pulse width	10	40 *(1)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high		1 *(1)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	45		µs
t_{ST2CK}	nSTATUS high to first rising edge on DCLK	1		µs
t_{DSU}	Data setup time before rising edge on DCLK	10		ns
t_{DH}	Data hold time after rising edge on DCLK	0		ns
t_{CH2B}	First rising DCLK to first rising RDYnBSY *(2)*	0.75 *(3)*		µs

Table 8–11. PPS Timing Parameters for Mercury Devices (Part 2 of 2)

Symbol	Parameter	Min	Max	Units
t_{CH}	DCLK high time	10		ns
t_{CL}	DCLK low time	10		ns
t_{CLK}	DCLK period	20		ns
f_{MAX}	DCLK frequency		50	MHz
t_{CD2UM}	CONF_DONE high to user mode *(4)*	6	28	µs

Notes to Table 8–11:

(1) This value is obtainable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(2) The RDYnBSY pin is not required in the PPS mode. Configuration data can be sent every 8 DCLK cycles without monitoring this status pin.

(3) This parameter depends on the DCLK frequency. The RDYnBSY signal goes high 7.5 clock cycles after the rising edge of DCLK. This value was calculated with a DCLK frequency of 10 MHz.

(4) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device. If the clock source is CLKUSR, multiply the clock period by 136 to obtain this value.

Table 8–12. PPS Timing Parameters for APEX 20K Devices (Part 1 of 2)

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		200	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		200	ns
t_{CFG}	nCONFIG low pulse width	8		µs
t_{STATUS}	nSTATUS low pulse width	10	40 *(1)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high		1 *(1)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	40		µs
t_{ST2CK}	nSTATUS high to first rising edge on DCLK	1		µs
t_{DSU}	Data setup time before rising edge on DCLK	10		ns
t_{DH}	Data hold time after rising edge on DCLK	0		ns
t_{CH2B}	First rising DCLK to first rising RDYnBSY *(2)*	0.75 *(3)*		µs
t_{CH}	DCLK high time	15		ns
t_{CL}	DCLK low time	15		ns
t_{CLK}	DCLK period	30		ns
f_{MAX}	DCLK frequency		33.3	MHz

Table 8–12. PPS Timing Parameters for APEX 20K Devices (Part 2 of 2)

Symbol	Parameter	Min	Max	Units
t_{CD2UM}	CONF_DONE high to user mode *(4)*	2	8	µs

Notes to Tables 8–12:

(1) This value is obtainable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(2) The RDYnBSY pin is not required in the PPS mode. Configuration data can be sent every 8 DCLK cycles without monitoring this status pin.

(3) This parameter depends on the DCLK frequency. The RDYnBSY signal goes high 7.5 clock cycles after the rising edge of DCLK. This value was calculated with a DCLK frequency of 10 MHz.

(4) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device. If the clock source is CLKUSR, multiply the clock period by 40 to obtain this value.

Table 8–13. PPS Timing Parameters for ACEX 1K & FLEX 10KE Devices

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		200	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		200	ns
t_{CFG}	nCONFIG low pulse width	2		µs
t_{STATUS}	nSTATUS low pulse width	1	10 *(1)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high		4 *(1)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	5		µs
t_{ST2CK}	nSTATUS high to first rising edge on DCLK	1		µs
t_{DSU}	Data setup time before rising edge on DCLK	10		ns
t_{DH}	Data hold time after rising edge on DCLK	0		ns
t_{CH2B}	First rising DCLK to first rising RDYnBSY *(2)*	0.75 *(3)*		µs
t_{CH}	DCLK high time	15		ns
t_{CL}	DCLK low time	15		ns
t_{CLK}	DCLK period	30		ns
f_{MAX}	DCLK frequency		33.3	MHz
t_{CD2UM}	CONF_DONE high to user mode *(4)*	0.6	2	µs

Notes to Table 8–13:

(1) This value is obtainable if users do not delay configuration by extending the nSTATUS low pulse width.

(2) The RDYnBSY pin is not required in the PPS mode. Configuration data can be sent every 8 DCLK cycles without monitoring this status pin.

(3) This parameter depends on the DCLK frequency. The RDYnBSY signal goes high 7.5 clock cycles after the rising edge of DCLK. This value was calculated with a DCLK frequency of 10 MHz.

(4) The minimum and maximum numbers apply only if DCLK is chosen as the clock source for starting up the device. If the clock source is CLKUSR, multiply the clock period by 10 to obtain this value.

Table 8–14. PPS Timing Parameters for FLEX 10K Devices

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		200	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		200	ns
t_{CFG}	nCONFIG low pulse width	2		µs
t_{STATUS}	nSTATUS low pulse width	1	10 *(1)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high		4 *(1)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	5		µs
t_{ST2CK}	nSTATUS high to first rising edge on DCLK	1		µs
t_{DSU}	Data setup time before rising edge on DCLK	10		ns
t_{DH}	Data hold time after rising edge on DCLK	0		ns
t_{CH2B}	First rising DCLK to first rising RDYnBSY *(2)*	0.75 *(3)*		µs
t_{CH}	DCLK high time	30		ns
t_{CL}	DCLK low time	30		ns
t_{CLK}	DCLK period	60		ns
f_{MAX}	DCLK frequency		16.7	MHz
t_{CD2UM}	CONF_DONE high to user mode *(4)*	0.6	2	µs

Notes to Table 8–14:

(1) This value is obtainable if users do not delay configuration by extending the nSTATUS low pulse width.

(2) The RDYnBSY pin is not required in the PPS mode. Configuration data can be sent every 8 DCLK cycles without monitoring this status pin.

(3) This parameter depends on the DCLK frequency. The RDYnBSY signal goes high 7.5 clock cycles after the rising edge of DCLK. This value was calculated with a DCLK frequency of 10 MHz.

(4) The minimum and maximum numbers apply only if DCLK is chosen as the clock source for starting up the device. If the clock source is CLKUSR, multiply the clock period by 10 to obtain this value.

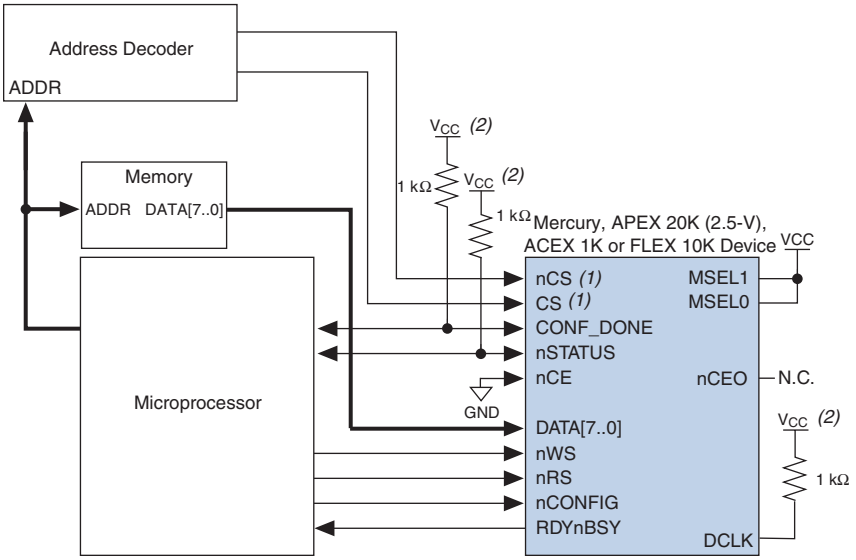
Device configuration options and how to create configuration files are discussed further in *Software Settings*, chapters 6 and 7 in volume 2 of the *Configuration Handbook*.

Passive Parallel Asynchronous Configuration

Passive Parallel Asynchronous (PPA) configuration uses an intelligent host, such as a microprocessor, to transfer configuration data from a storage device, such as flash memory, to the target Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices. Configuration data can be stored in TTF, RBF, or HEX format. The host system outputs byte-wide data and the accompanying strobe signals to the FPGA. When using PPA, you should pull the DCLK pin high through a 1-kΩ pull-up resistor to prevent unused configuration input pins from floating.

Figure 8–20 shows the configuration interface connections between the FPGA and a microprocessor for single device PPA configuration. The microprocessor or an optional address decoder can control the device's chip select pins, nCS and CS. The address decoder allows the microprocessor to select the Mercury, APEX 20K (2.5 V), ACEX 1K, or FLEX 10K device by accessing a particular address, which simplifies the configuration process. The nCS and CS pins must be held active during configuration and initialization.

Figure 8–20. Single Device PPA Configuration Using a Microprocessor *Note (1)*



Notes to Figure 8–20:

(1) If not used, the CS pin can be connected to V_{CC} directly. If not used, the nCS pin can be connected to GND directly.

(2) The pull-up resistor should be connected to a supply that provides an acceptable input signal for the device.

During PPA configuration, it is only required to use either the nCS or CS pin. Therefore, if only one chip-select input is used, the other must be tied to the active state. For example, nCS can be tied to ground while CS is toggled to control configuration. The device's nCS or CS pins can be toggled during PPA configuration if the design meets the specifications set for t_{CSSU}, t_{WSP}, and t_{CSH} listed in Tables 8–15 through 8–17.

Upon power-up, the Mercury, APEX 20K (2.5 V), ACEX 1K, or FLEX 10K device goes through a Power-On Reset (POR) for approximately 5 µs. During POR, the device resets and holds nSTATUS low, and tri-states all user I/O pins. Once the FPGA successfully exits POR, all user I/O pins

are tri-stated. Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10KE devices have weak pull-up resistors on the user I/O pins which are on before and during configuration.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the Operating Conditions table of the appropriate device family data sheet.

The configuration cycle consists of 3 stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration, the microprocessor must generate a low-to-high transition on the nCONFIG pin.

VCCINT and VCCIO pins on the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 1-kΩ pull-up resistor. Once nSTATUS is released the FPGA is ready to receive configuration data and the configuration stage begins. When nSTATUS is pulled high, the microprocessor should then assert the target device's nCS pin low and/or CS pin high. Next, the microprocessor places an 8-bit configuration word (one byte) on the target device's DATA[7..0] pins and pulses the nWS pin low.

On the rising edge of nWS, the target device latches in a byte of configuration data and drives its RDYnBSY signal low, which indicates it is processing the byte of configuration data. The microprocessor can then perform other system functions while the Mercury, APEX 20K (2.5 V), ACEX 1K, or FLEX 10K device is processing the byte of configuration data.

During the time RDYnBSY is low, the Mercury, APEX 20K (2.5 V), ACEX 1K, or FLEX 10K device internally processes the configuration data using its internal oscillator (typically 10 MHz). When the device is ready for the next byte of configuration data, it will drive RDYnBSY high. If the microprocessor senses a high signal when it polls RDYnBSY, the microprocessor sends the next byte of configuration data to the FPGA.

Alternatively, the nRS signal can be strobed low, causing the RDYnBSY signal to appear on DATA7. Because RDYnBSY does not need to be monitored, this pin doesn't need to be connected to the microprocessor. Data should not be driven onto the data bus while nRS is low because it will cause contention on the DATA7 pin. If the nRS pin is not used to monitor configuration, it should be tied high.

To simplify configuration and save an I/O port, the microprocessor can wait for the total time of t_{BUSY}(max) + t_{RDY2WS} + t_{W2SB} before sending the next data byte. In this set-up, nRS should be tied high and RDYnBSY does not need to be connected to the microprocessor. The t_{BUSY}, t_{RDY2WS} and t_{W2SB} timing specifications are listed in Tables 8–15 through 8–17.

Next, the microprocessor checks nSTATUS and CONF_DONE. If nSTATUS is not low and CONF_DONE is not high, the microprocessor sends the next data byte. However, if nSTATUS is not low and all the configuration data has been received, the device is ready for initialization. After the FPGA has received all configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by an external 1-kΩ pull-up resistor. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin.

In Mercury and APEX 20K (2.5 V) devices, the initialization clock source is either the FPGA's internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Mercury or APEX 20K (2.5 V) device allows enough clock cycles for proper initialization. Therefore, sending the entire configuration file to the device is sufficient to configure and initialize the device.

In ACEX 1K and FLEX 10K devices, the initialization clock source is either an external host (e.g. a configuration device or microprocessor) or the optional CLKUSR pin. By default, in PPA, the device uses its internal oscillator (typically 10MHz) to clock the initialization cycle. The ACEX 1K or FLEX 10K device will take care to provide itself with enough clock cycles for proper initialization.

You also have the flexibility to synchronize initialization of multiple devices by using the *CLKUSR* option. The *Enable user-supplied start-up clock (CLKUSR)* option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR will not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, Mercury devices require 136 clock cycles to initialize properly, APEX 20K (2.5 V) devices require 40 clock cycles, ACEX and FLEX 10K devices require 10 clock cycles.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. This *Enable INIT_DONE output* option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used it will be high due to an external 1-kΩ pull-up when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device

(during the first frame of configuration data), the INIT_DONE pin will go low. When initialization is complete, the INIT_DONE pin will be released and pulled high. The microprocessor must be able to detect this low-to-high transition which signals the FPGA has entered user mode. In user-mode, the user I/O pins will no longer have weak pull-up resistors and will function as assigned in your design. When initialization is complete, the FPGA enters user mode.

To ensure DATA0 is not left floating at the end of configuration, the microprocessor must take care to drive them either high or low, whichever is convenient on your board. After configuration, the nCS, CS, nRS, nWS, RDYnBSY, and DATA[7..1] pins can be used as user I/O pins. When the PPA scheme is chosen in the Quartus II software, as a default these I/O pins are tri-stated in user mode and should be driven by the microprocessor. To change this default option in the Quartus II software, select the **Dual-Purpose Pins** tab of the **Device & Pin Options** dialog box.

If an error occurs during configuration, the FPGA drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the microprocessor that there is an error. If the *Auto-Restart Configuration After Error* option-available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box-is turned on, the FPGA releases nSTATUS after a reset time-out period (maximum of 40 µs). After nSTATUS is released and pulled high by a pull-up resistor, the microprocessor can try to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition on nCONFIG to restart the configuration process.

The microprocessor can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The CONF_DONE pin must be monitored by the microprocessor to detect errors and determine when programming completes. If the microprocessor sends all configuration data but CONF_DONE or INIT_DONE has not gone high, the microprocessor must reconfigure the target device.

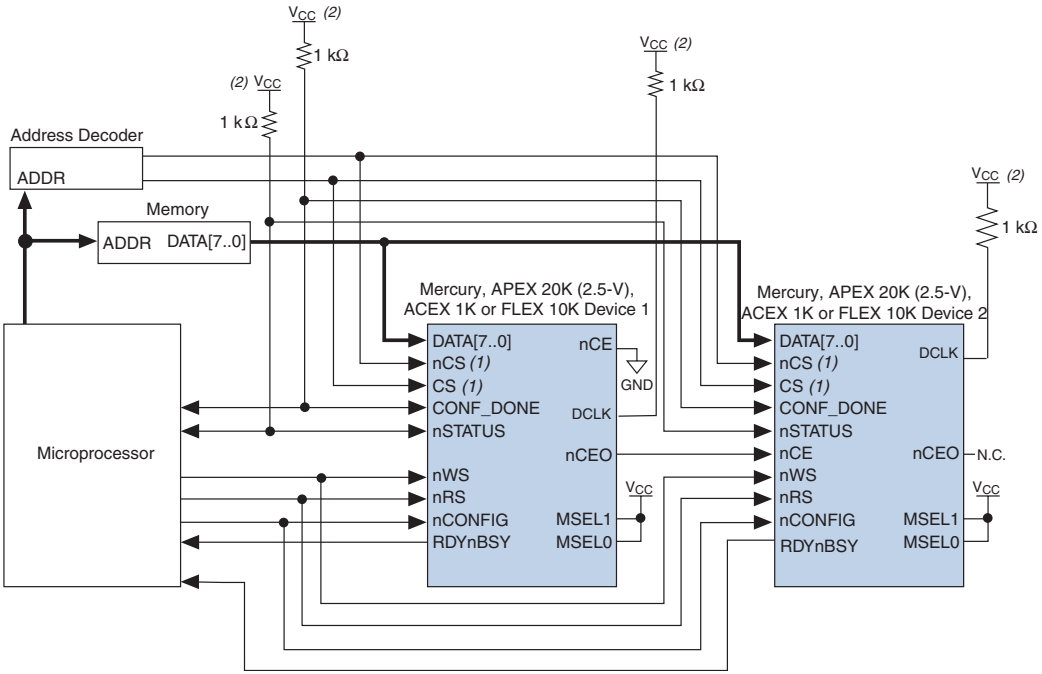
If the optional CLKUSR pin is being used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure CLKUSR continues toggling during the time nSTATUS is low (maximum of 40 µs).

When the FPGA is in user-mode, a reconfiguration can be initiated by transitioning the nCONFIG pin low-to-high. The nCONFIG pin should be low for at least 21 µs for Mercury devices, 8 µs for APEX 20K devices, and 2 µs for ACEX 1K and FLEX 10K devices. When nCONFIG is pulled low,

the FPGA also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Once nCONFIG returns to a logic high state and nSTATUS is released by the FPGA, reconfiguration begins.

Figure 8–21 shows how to configure multiple Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices using a microprocessor. This circuit is similar to the PPA configuration circuit for a single device, except the devices are cascaded for multi-device configuration.

Figure 8–21. Multi-Device PPA Configuration Using a Microprocessor



Notes to Figure 8–21:

(1) If not used, the CS pin can be connected to V_{CC} directly. If not used, the nCS pin can be connected to GND directly.

(2) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.

In multi-device PPA configuration the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin

configuration. The second device in the chain begins configuration within one clock cycle; therefore, the transfer of data destinations is transparent to the microprocessor.

Each device's RDYnBSY pin can have a separate input to the microprocessor. Alternatively, if the microprocessor is pin limited, all the RDYnBSY pins can feed an AND gate and the output of the AND gate can feed the microprocessor. For example, if you have 2 devices in a PPA configuration chain, the second device's RDYnBSY pin will be high during the time that the first device is being configured. When the first device has been successfully configured, it will driven nCEO low to activate the next device in the chain and drive its RDYnBSY pin high. Therefore, since RDYnBSY signal is driven high before configuration and after configuration before entering user-mode, the device being configured will govern the output of the AND gate.

The nRS signal can be used in multi-device PPA chain since the Mercury, APEX 20K (2.5 V), ACEX 1K, or FLEX 10K device will tri-state its DATA[7..0] pins before configuration and after configuration before entering user-mode to avoid contention. Therefore, only the device that is currently being configured will respond to the nRS strobe by asserting DATA7.

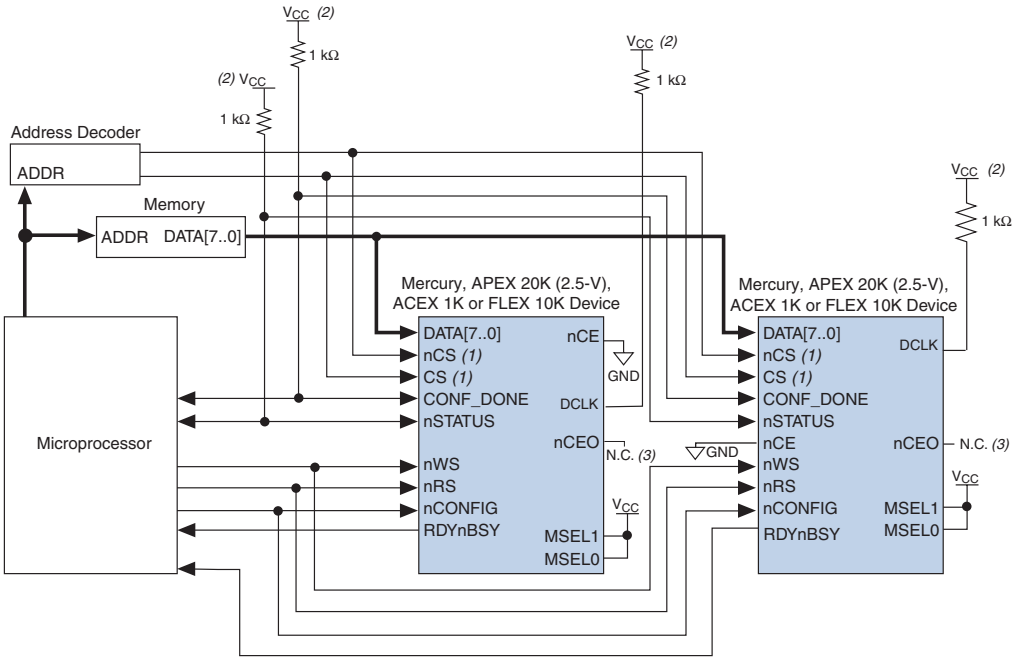
All other configuration pins (nCONFIG, nSTATUS, DATA[7..0], nCS, CS, nWS, nRS and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they may require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Since all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first FPGA flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single FPGA detecting an error.

If the *Auto-Restart Configuration After Error* option is turned on, the FPGAs release their nSTATUS pins after a reset time-out period (maximum of 40 µs). After all nSTATUS pins are released and pulled high, the microprocessor can try to reconfigure the chain without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition on nCONFIG to restart the configuration process.

In your system, you may have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DATA[7..1], nCS, CS, nWS, nRS and CONF_DONE) are connected to every device in the chain. You should pay special attention to the configuration signals because they may require buffering to ensure signal integrity and prevent clock skew problems. Specifically, ensure that the DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete configuration at the same time. Figure 8–22 shows multi-device PPA configuration when both devices are receiving the same configuration data.

Figure 8–22. Multiple-Device PPA Configuration Using a Microprocessor When Both FPGAs Receive the Same Data



Notes to Figure 8–22:

(1) If not used, the CS pin can be connected to V_{CC} directly. If not used, the nCS pin can be connected to GND directly.

(2) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.

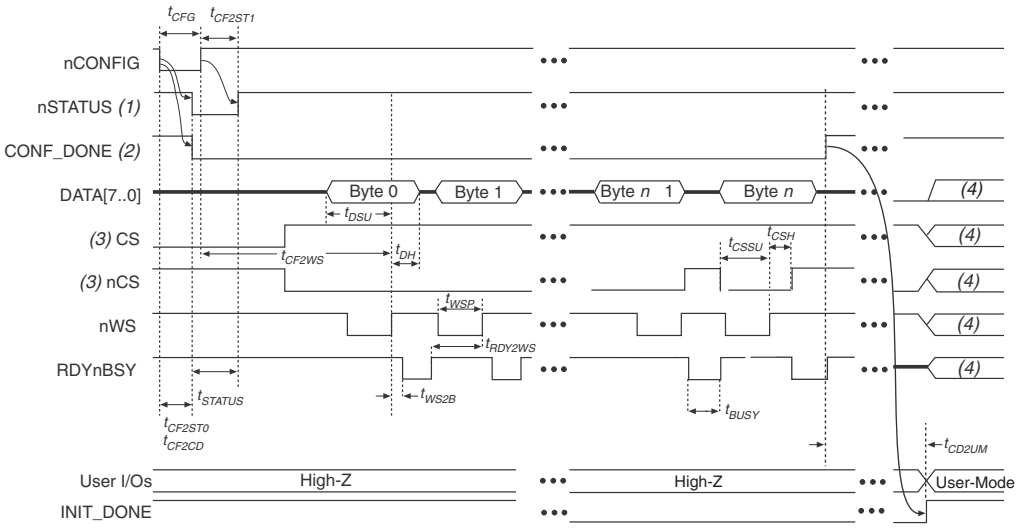
(3) The nCEO pins of both devices are left unconnected when configuring the same configuration data into multiple devices.

You can use a single configuration chain to configure Mercury, APEX 20K, ACEX, and FLEX 10KE devices with other Altera devices that support PPA configuration, such as Stratix® or APEX II devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information on configuring multiple Altera devices in the same configuration chain, refer to *Configuring Mixed Altera FPGA Chains* chapter 8 of volume 2 in the *Configuration Handbook*.

Figure 8–23 shows the timing waveform for the PPA configuration scheme using a microprocessor.

Figure 8–23. PPA Configuration Timing Waveform



Notes to Figure 8–23:

(1) Upon power-up, the Mercury, APEX 20K, ACEX 1K, or FLEX 10K device holds nSTATUS low for not more than 5 µs after V_{CC} reaches its minimum requirement.

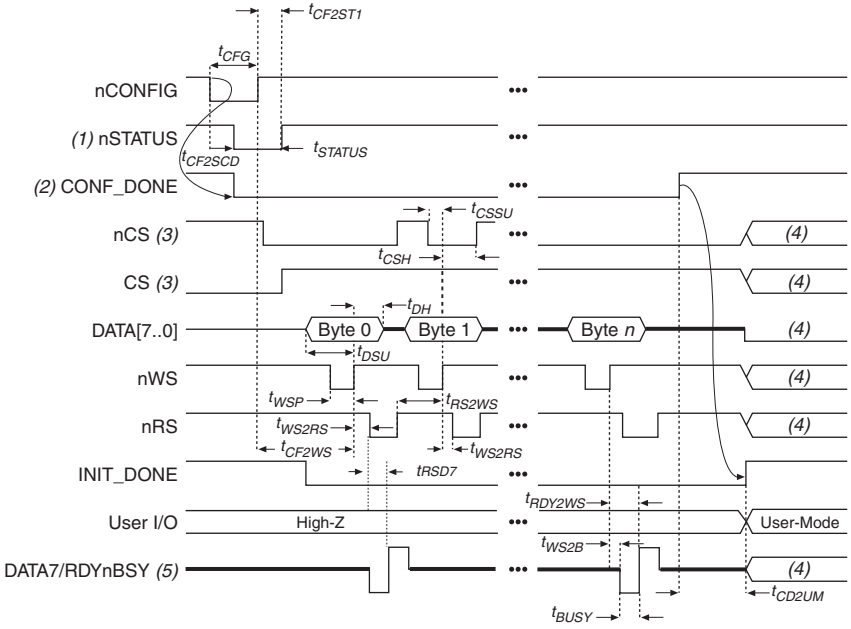
(2) Upon power-up, before and during configuration, CONF_DONE is low.

(3) The user can toggle nCS or CS during configuration if the design meets the specification for t_{CSSU}, t_{WSP}, and t_{CSH}.

(4) DATA0 should not be left floating after configuration. It should be driven high or low, whichever is more convenient. DATA[7..1], CS, nCS, nWS, nRS and RDYnBSY are available as user I/O pins after configuration and the state of theses pins depends on the dual-purpose pin settings.

Figure 8–24 shows the timing waveform for the PPA configuration scheme when using a strobed nRS and nWS signal.

Figure 8–24. PPA Configuration Timing Waveform Using nRS & nWS



Notes to Figure 8–24:

(1) Upon power-up, the Mercury, APEX 20K, ACEX 1K, or FLEX 10K device holds nSTATUS low for not more than 5 µs after V_{CC} reaches its minimum requirement.

(2) Upon power-up, before and during configuration, CONF_DONE is low.

(3) The user can toggle nCS or CS during configuration if the design meets the specification for t_{CSSU}, t_{WSP}, and t_{CSH}.

(4) DATA0 should not be left floating after configuration. It should be driven high or low, whichever is more convenient. DATA[7..1], CS, nCS, nWS, nRS, and RDYnBSY are available as user I/O pins after configuration and the state of theses pins depends on the dual-purpose pin settings.

(5) DATA7 is a bidirectional pin. It is an input for configuration data input, but it is an output to show the status of RDYnBSY.

Tables 8–15 through 8–17 define the timing parameters for Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices for PPA configuration.

Table 8–15. PPA Timing Parameters for Mercury Devices (Part 1 of 2) *Note (1)*

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		200	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		200	ns
t_{CFG}	nCONFIG low pulse width	21		µs

Table 8–15. PPA Timing Parameters for Mercury Devices (Part 2 of 2) *Note (1)*

Symbol	Parameter	Min	Max	Units
t_{STATUS}	nSTATUS low pulse width	10	40 *(2)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high		1 *(2)*	µs
t_{CSSU}	Chip select setup time before rising edge on nWS	10		ns
t_{CSH}	Chip select hold time after rising edge on nWS	0		ns
t_{CF2WS}	nCONFIG high to first rising edge on nWS	40		µs
t_{DSU}	Data setup time before rising edge on nWS	10		ns
t_{DH}	Data hold time after rising edge on nWS	0		ns
t_{WSP}	nWS low pulse width	200		ns
t_{WS2B}	nWS rising edge to RDYnBSY low		50	ns
t_{BUSY}	RDYnBSY low pulse width	0.4	1.6	µs
t_{RDY2WS}	RDYnBSY rising edge to nWS rising edge	50		ns
t_{WS2RS}	nWS rising edge to nRS falling edge	200		ns
t_{RS2WS}	nRS rising edge to nWS rising edge	200		ns
t_{RSD7}	nRS falling edge to DATA7 valid with RDYnBSY signal		50	ns
t_{CD2UM}	CONF_DONE high to user mode *(3)*	6	28	µs

Notes to Table 8–15:

(1) This information is preliminary.

(2) This value is obtainable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(3) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device. If the clock source is CLKUSR, multiply the clock period by 136 to obtain this value.

Table 8–16. PPA Timing Parameters for APEX 20K Devices (Part 1 of 2)

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		200	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		200	ns
t_{CFG}	nCONFIG low pulse width	8		µs
t_{STATUS}	nSTATUS low pulse width	10	40 *(1)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high		1 *(1)*	µs
t_{CSSU}	Chip select setup time before rising edge on nWS	10		ns
t_{CSH}	Chip select hold time after rising edge on nWS	0		ns
t_{CF2WS}	nCONFIG high to first rising edge on nWS	40		µs
t_{DSU}	Data setup time before rising edge on nWS	10		ns

Table 8–16. PPA Timing Parameters for APEX 20K Devices (Part 2 of 2)

Symbol	Parameter	Min	Max	Units
t_{DH}	Data hold time after rising edge on nWS	0		ns
t_{WSP}	nWS low pulse width	200		ns
t_{WS2B}	nWS rising edge to RDYnBSY low		50	ns
t_{BUSY}	RDYnBSY low pulse width	0.1	1.6	µs
t_{RDY2WS}	RDYnBSY rising edge to nWS rising edge	50		ns
t_{WS2RS}	nWS rising edge to nRS falling edge	200		ns
t_{RS2WS}	nRS rising edge to nWS rising edge	200		ns
t_{RSD7}	nRS falling edge to DATA7 valid with RDYnBSY signal		50	ns
t_{CD2UM}	CONF_DONE high to user mode *(2)*	2	8	µs

Notes to Table 8–16:

(1) This value is obtainable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(2) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device. If the clock source is CLKUSR, multiply the clock period by 40 to obtain this value.

Table 8–17. PPA Timing Parameters for ACEX 1K & FLEX 10K Devices (Part 1 of 2)

Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		200	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		200	ns
t_{CFG}	nCONFIG low pulse width	2		µs
t_{STATUS}	nSTATUS low pulse width	1	10 *(1)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high		4 *(1)*	µs
t_{CSSU}	Chip select setup time before rising edge on nWS	20		ns
t_{CSH}	Chip select hold time after rising edge on nWS	10 *(2)* 15 *(3)*		ns
t_{CF2WS}	nCONFIG high to first rising edge on nWS	5		µs
t_{DSU}	Data setup time before rising edge on nWS	20		ns
t_{DH}	Data hold time after rising edge on nWS	0		ns
t_{WSP}	nWS low pulse width	200		ns
t_{WS2B}	nWS rising edge to RDYnBSY low		50	ns
t_{BUSY}	RDYnBSY low pulse width	0.4	1.6	µs
t_{RDY2WS}	RDYnBSY rising edge to nWS rising edge	50		ns
t_{WS2RS}	nWS rising edge to nRS falling edge	200		ns

Table 8–17. PPA Timing Parameters for ACEX 1K & FLEX 10K Devices (Part 2 of 2)

Symbol	Parameter	Min	Max	Units
t_{RS2WS}	nRS rising edge to nWS rising edge	200		ns
t_{RSD7}	nRS falling edge to DATA7 valid with RDYnBSY signal		50	ns
t_{CD2UM}	CONF_DONE high to user mode *(4)*	0.6	2	µs

Notes to Table 8–17:

(1) This value is obtainable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(2) This parameter value applies to EPF10K10, EPF10K20, EPF10K40, EPF10K50, all FLEX 10KA, and FLEX 10KE devices.

(3) This parameter value applies to only EPF10K70 and EPF10K100 devices.

(4) If the clock source is CLKUSR, multiply the clock period by 10 to obtain this value.

Device configuration options and how to create configuration files are discussed further in *Software Settings*, chapter 6 and 7 in volume 2 of the *Configuration Handbook*.

JTAG Configuration

The Joint Test Action Group (JTAG) has developed a specification for boundary-scan testing. This boundary-scan test (BST) architecture offers the capability to efficiently test components on PCBs with tight lead spacing. The BST architecture can test pin connections without using physical test probes and capture functional data while a device is operating normally. The JTAG circuitry can also be used to shift configuration data into the device.

For more information on JTAG boundary-scan testing, refer to *Application Note 39: IEEE 1149.1 (JTAG) Boundary-Scan Testing* in *Altera Devices*.

A device operating in JTAG mode uses four required pins, TDI, TDO, TMS, and TCK, and one optional pin, TRST. All user I/O pins are tri-stated during JTAG configuration. Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices are designed such that JTAG instructions have precedence over any device configuration modes. This means that JTAG configuration can take place without waiting for other configuration modes to complete. For example, if you attempt JTAG configuration of Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K FPGAs during PS configuration, PS configuration will be terminated and JTAG configuration will begin.

Table 8–18 explains each JTAG pin's function.

Table 8–18. JTAG Pin Descriptions

Pin	Description	Function
TDI	Test data input	Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}.
TDO	Test data output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by leaving this pin unconnected.
TMS	Test mode select	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}.
TCK	Test clock input	The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND.
TRST *(1)*	Test reset input (optional)	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to IEEE Std. 1149.1. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND.

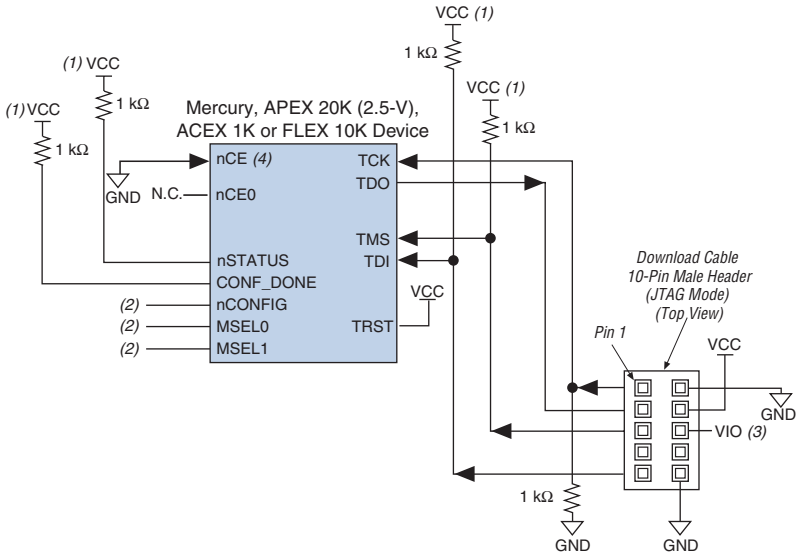
Note to Table 8–18:

(1) FLEX 10K devices in 144-pin thin quad flat pack (TQFP) packages do not have a TRST pin. Therefore, the TRST pin can be ignored when using these devices.

If V_{CCIO} of the bank where the JTAG pins reside, are tied to 3.3-V, both the I/O pins and JTAG TDO port will drive at 3.3-V levels.

During JTAG configuration, data can be downloaded to the device on the PCB through the USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV header. Configuring devices through a cable is similar to programming devices in-system, except the TRST pin should be connected to V_{CC}. This ensures that the TAP controller is not reset. Figure 8–25. shows JTAG configuration of a single Mercury, APEX 20K (2.5 V), ACEX 1K, or FLEX 10K device.

Figure 8–25. JTAG Configuration of a Single Device Using a Download Cable



Notes to Figure 8–25:

(1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster (VIO pin), ByteBlaster II, or ByteBlasterMV cable.

(2) The nCONFIG, MSEL0, and MSEL1 pins should be connected to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect nCONFIG to V_{CC}, and MSEL0 and MSEL1 to ground.

(3) Pin 6 of the header is a VIO reference voltage for the MasterBlaster output driver. VIO should match the device's V_{CCIO}. Refer to the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value. In the ByteBlasterMV, this pin is a no connect. In the USB Blaster and ByteBlaster II, this pin is connected to nCE when it is used for Active Serial programming, otherwise it is a no connect.

(4) nCE must be connected to GND or driven low for successful JTAG configuration.

To configure a single device in a JTAG chain, the programming software places all other devices in BYPASS mode. In BYPASS mode, devices pass programming data from the TDI pin to the TDO pin through a single bypass register without being affected internally. This scheme enables the programming software to program or verify the target device. Configuration data driven into the device appears on the TDO pin one clock cycle later.

Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices have dedicated JTAG pins that always function as JTAG pins. JTAG testing can be performed on Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices both before and after configuration, but not during configuration. The chip-wide reset (DEV_CLRn) and chip-wide output enable (DEV_OE) pins on Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices do not affect JTAG boundary-scan or programming operations. Toggling these pins does not affect JTAG operations (other than the usual boundary-scan operation).

When designing a board for JTAG configuration of Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices, the dedicated configuration pins should be considered. Table 8–19 shows how these pins should be connected during JTAG configuration.

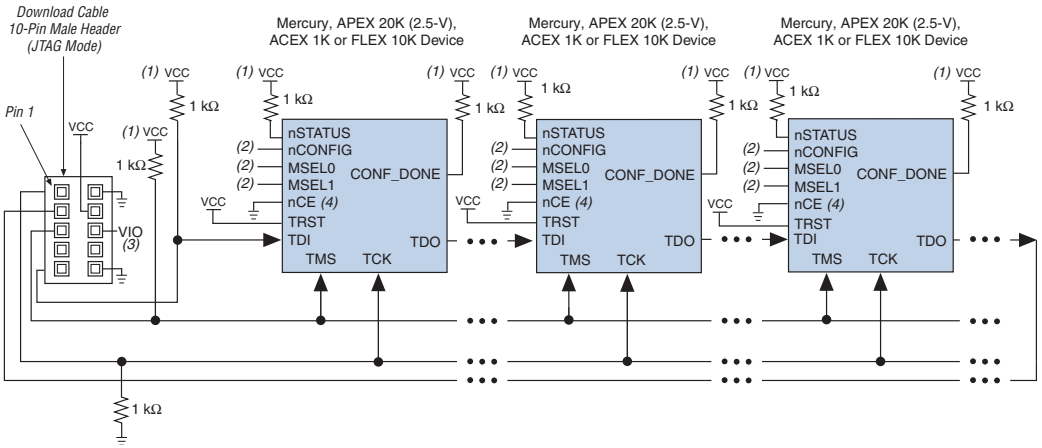
Table 8–19. Dedicated Configuration Pin Connections During JTAG Configuration

Signal	Description
nCE	On all Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices in the chain, nCE should be driven low by connecting it to ground, pulling it low via a resistor, or driving it by some control circuitry. For devices that are also in multi-device PS, PPS or PPA configuration chains, the nCE pins should be connected to GND during JTAG configuration or JTAG configured in the same order as the configuration chain.
nCEO	On all Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K devices in the chain, nCEO can be left floating or connected to the nCE of the next device. See nCE description above.
MSEL	These pins must not be left floating. These pins support whichever non-JTAG configuration is used in production. If only JTAG configuration is used, you should tie both pins to ground.
nCONFIG	Driven high by connecting to V_{CC}, pulling high via a resistor, or driven by some control circuitry.
nSTATUS	Pull to V_{CC} via a 1-kΩ resistor. When configuring multiple devices in the same JTAG chain, each nSTATUS pin should be pulled up to V_{CC} individually. nSTATUS pulling low in the middle of JTAG configuration indicates that an error has occurred.
CONF_DONE	Pull to V_{CC} via a 1-kΩ resistor. When configuring multiple devices in the same JTAG chain, each CONF_DONE pin should be pulled up to V_{CC} individually. CONF_DONE going high at the end of JTAG configuration indicates successful configuration.
DCLK	Should not be left floating. Drive low or high, whichever is more convenient on your board.
DATA0	Should not be left floating. Drive low or high, whichever is more convenient on your board.

When programming a JTAG device chain, one JTAG-compatible header is connected to several devices. The number of devices in the JTAG chain is limited only by the drive capability of the download cable. When four or more devices are connected in a JTAG chain, Altera recommends buffering the TCK, TDI, and TMS pins with an on-board buffer.

JTAG-chain device programming is ideal when the system contains multiple devices, or when testing your system using JTAG BST circuitry. Figure 8–26 shows multi-device JTAG configuration.

Figure 8–26. JTAG Configuration of Multiple Devices Using a Download Cable



Notes to Figure 8–26:

(1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster (VIO pin), ByteBlaster II or ByteBlasterMV cable.

(2) The nCONFIG, MSEL0, and MSEL1 pins should be connected to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect nCONFIG to V_{CC}, and MSEL0 and MSEL1 to ground.

(3) Pin 6 of the header is a VIO reference voltage for the MasterBlaster output driver. VIO should match the device's V_{CCIO}. Refer to the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value. In the ByteBlasterMV, this pin is a no connect. In the USB Blaster and ByteBlaster II, this pin is connected to nCE when it is used for Active Serial programming, otherwise it is a no connect.

(4) nCE must be connected to GND or driven low for successful JTAG configuration.

The nCE pin must be connected to GND or driven low during JTAG configuration. In multi-device PS, PPS, and PPA configuration chains, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. Therefore, if these devices are also in a JTAG chain, you should make sure the nCE pins are connected to GND during JTAG configuration or that the devices are JTAG configured in the same order as the configuration chain. As long as the devices are JTAG configured in the same order as the multi-device configuration chain, the nCEO of the previous device will drive nCE of the next device low when it has successfully been JTAG configured.

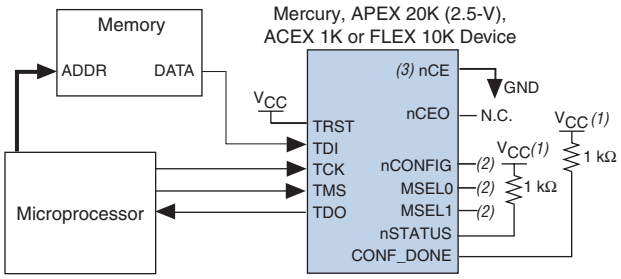
Other Altera devices that have JTAG support can be placed in the same JTAG chain for device programming and configuration.

For more information about configuring multiple Altera devices in the same configuration chain, refer to *Configuring Mixed Altera FPGA Chains* chapter 8 in volume 2 of the *Configuration Handbook*.

The Quartus II software verifies successful JTAG configuration upon completion. At the end of configuration, the software checks the state of CONF_DONE through the JTAG port. If CONF_DONE is not high, the Quartus II software indicates that configuration has failed. If CONF_DONE is high, the software indicates that configuration was successful. When Quartus II generates a JAM file for a multi-device chain, it contains instructions so that all the devices in the chain will be initialized at the same time.

Figure 8–27 shows JTAG configuration of a Mercury, APEX 20K (2.5 V), ACEX 1K, or FLEX 10K FPGA with a microprocessor.

Figure 8–27. JTAG Configuration of a Single Device Using a Microprocessor



Notes to Figure 8–27:

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain.

(2) Connect the nCONFIG, MSEL1, and MSEL0 pins to support a non-JTAG configuration scheme. If your design only uses JTAG configuration, connect the nCONFIG pin to V_{CC} and the MSEL1 and MSEL0 pins to ground.

(3) nCE must be connected to GND or driven low for successful JTAG configuration.

Jam STAPL

Jam STAPL, JEDEC standard JESD-71, is a standard file format for in-system programmability (ISP) purposes. Jam STAPL supports programming or configuration of programmable devices and testing of electronic systems, using the IEEE 1149.1 JTAG interface. Jam STAPL is a freely licensed open standard.

The Jam Player provides an interface for manipulating the IEEE Std. 1149.1 JTAG TAP state machine.

For more information on JTAG and Jam STAPL in embedded environments, refer to *AN 122: Using Jam STAPL for ISP & ICR via an Embedded Processor*. To download the jam player, visit the Altera web site at:

www.altera.com/support/software/download/programming/jam/jam-index.jsp

Configuring Mercury, APEX 20K (2.5 V), ACEX 1K & FLEX 10K FPGAs with JRunner

JRunner is a software driver that allows you to configure Altera FPGAs, including Mercury, APEX 20K (2.5 V), ACEX 1K, and FLEX 10K FPGAs, through the ByteBlaster II or ByteBlasterMV cables in JTAG mode. The programming input file supported is in RBF format. JRunner also requires a Chain Description File (**.cdf**) generated by the Quartus II software. JRunner is targeted for embedded JTAG configuration. The source code has been developed for the Windows NT operating system (OS). You can customize the code to make it run on other platforms.

For more information on the JRunner software driver, refer to the *JRunner Software Driver: An Embedded Solution to the JTAG Configuration White Paper* and the source files.

Device Configuration Pins

The following tables describe the connections and functionality of all the configuration related pins on the Mercury, APEX 20K (2.5 V), ACEX 1K, or FLEX 10K device. Table 8–20 describes the dedicated configuration pins, which are required to be connected properly on your board for successful configuration. Some of these pins may not be required for your configuration schemes.

Table 8–20. Dedicated Configuration Pins (Part 1 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
MSEL0 MSEL1	N/A	All	Input	Two-bit configuration input that sets the Mercury, APEX 20K (2.5 V), ACEX 1K, or FLEX 10K device configuration scheme. See Table 8–6 for the appropriate connections. These pins must remain at a valid state during power-up, before nCONFIG is pulled low to initiate a reconfiguration and during configuration.
VCCSEL	N/A	All	Input	This pin is only available in Mercury devices. Dedicated input that ensures the configuration related I/O banks have powered up to the appropriate 1.8-V or 2.5-V/3.3-V voltage levels before starting configuration. A logic high (1.5 V, 1.8 V, 2.5 V, 3.3 V) means 2.5 V/3.3 V, and a logic low means 1.8 V.
nCONFIG	N/A	All	Input	Configuration control input. Pulling this pin low during user-mode will cause the FPGA to lose its configuration data, enter a reset state, tri-state all I/O pins, and returning this pin to a logic high level will initiate a reconfiguration. If your configuration scheme uses an enhanced configuration device or EPC2 device, nCONFIG can be tied directly to V_{CC} or to the configuration device's nINIT_CONF pin.

Table 8–20. Dedicated Configuration Pins (Part 2 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nSTATUS	N/A	All	Bidirectional open-drain	The FPGA drives nSTATUS low immediately after power-up and releases it within 5 µs. (When using a configuration device, the configuration device holds nSTATUS low for up to 200 ms.) Status output. If an error occurs during configuration, nSTATUS is pulled low by the target device. Status input. If an external source drives the nSTATUS pin low during configuration or initialization, the target device enters an error state. Driving nSTATUS low after configuration and initialization does not affect the configured device. If a configuration device is used, driving nSTATUS low will cause the configuration device to attempt to configure the FPGA, but since the FPGA ignores transitions on nSTATUS in user-mode, the FPGA will not reconfigure. To initiate a reconfiguration, nCONFIG must be pulled low. The enhanced configuration devices' and EPC2 devices' OE and nCS pins have optional internal programmable pull-up resistors. If internal pull-up resistors are used, external 1-kΩ pull-up resistors should not be used on these pins.
CONF_DONE	N/A	All	Bidirectional open-drain	Status output. The target FPGA drives the CONF_DONE pin low before and during configuration. Once all configuration data is received without error and the initialization cycle starts, the target device releases CONF_DONE. Status input. After all data is received and CONF_DONE goes high, the target device initializes and enters user mode. Driving CONF_DONE low after configuration and initialization does not affect the configured device. The enhanced configuration devices' and EPC2 devices' OE and nCS pins have optional internal programmable pull-up resistors. If internal pull-up resistors are used, external 1-kΩ pull-up resistors should not be used on these pins.
nCE	N/A	All	Input	Active-low chip enable. The nCE pin activates the device with a low signal to allow configuration. The nCE pin must be held low during configuration, initialization, and user mode. In single device configuration, it should be tied low. In multi-device configuration, nCE of the first device is tied low while its nCEO pin is connected to nCE of the next device in the chain. The nCE pin must also be held low for successful JTAG programming of the FPGA.

Table 8–20. Dedicated Configuration Pins (Part 3 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nCEO	N/A	All	Output	Output that drives low when device configuration is complete. In single device configuration, this pin is left floating. In multi-device configuration, this pin feeds the next device's nCE pin. The nCEO of the last device in the chain is left floating.
DCLK	N/A	Synchronous configuration schemes (PS and PPS)	Input	Clock input used to clock data from an external source into the target device. Data is latched into the FPGA on the rising edge of DCLK. In PPA mode, DCLK should be tied high to V_{CC} to prevent this pin from floating. After configuration, this pin is tri-stated. In schemes that use a configuration device, DCLK will be driven low after configuration is done. In schemes that use a control host, DCLK should be driven either high or low, whichever is more convenient. Toggling this pin after configuration does not affect the configured device.
DATA0	N/A	All	Input	Data input. In serial configuration modes, bit-wide configuration data is presented to the target device on the DATA0 pin. After configuration, EPC1 and EPC1441 devices tri-state this pin, while EPC2 devices drive this pin high. In schemes that use a control host, DATA0 should be driven either high or low, whichever is more convenient. Toggling this pin after configuration does not affect the configured device.
DATA[7..1]	I/O	Parallel configuration schemes (PPS and PPA)	Inputs	Data inputs. Byte-wide configuration data is presented to the target device on DATA[7..0]. In serial configuration schemes, they function as user I/O pins during configuration, which means they are tri-stated. After PPA or PPS configuration, DATA[7..1] are available as a user I/O pins and the state of these pin depends on the Dual-Purpose Pin settings.
DATA7	I/O	PPA	Bidirectional	In the PPA configuration scheme, the DATA7 pin presents the RDYnBSY signal after the nRS signal has been strobed low. In serial configuration schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, DATA7 is available as a user I/O and the state of this pin depends on the Dual-Purpose Pin settings.

Table 8–20. Dedicated Configuration Pins (Part 4 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nWS	I/O	PPA	Input	Write strobe input. A low-to-high transition causes the device to latch a byte of data on the DATA[7..0] pins. In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, nWS is available as a user I/O and the state of this pin depends on the Dual-Purpose Pin settings.
nRS	I/O	PPA	Input	Read strobe input. A low input directs the device to drive the RDYnBSY signal on the DATA7 pin. If the nRS pin is not used in PPA mode, it should be tied high. In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, nRS is available as a user I/O and the state of this pin depends on the Dual-Purpose Pin settings.
RDYnBSY	I/O	PPS and PPA	Output	Ready output. A high output indicates that the target device is ready to accept another data byte. A low output indicates that the target device is busy and not ready to receive another data byte. In PPS and PPA configuration schemes, this pin will drive out high after power-up, before configuration and after configuration before entering user-mode. In non-PPS and non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPS and PPA configuration, RDYnBSY is available as a user I/O and the state of this pin depends on the Dual-Purpose Pin settings.
nCS/CS	I/O	PPA	Input	Chip-select inputs. A low on nCS and a high on CS select the target device for configuration. The nCS and CS pins must be held active during configuration and initialization. During the PPA configuration mode, it is only required to use either the nCS or CS pin. Therefore, if only one chip-select input is used, the other must be tied to the active state. For example, nCS can be tied to ground while CS is toggled to control configuration. In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, nCS and CS are available as a user I/O pins and the state of these pins depends on the Dual-Purpose Pin settings.

Table 8–21 describes the optional configuration pins. If these optional configuration pins are not enabled in the Quartus II software, they are available as general-purpose user I/O pins. Therefore during configuration they function as user I/O pins, which means they are tri-stated with weak pull-up resistors.

Table 8–21. Optional Configuration Pins

Pin Name	User Mode	Pin Type	Description
CLKUSR	N/A if option is on. I/O if option is off.	Input	Optional user-supplied clock input. Synchronizes the initialization of one or more devices. This pin is enabled by turning on the *Enable user-supplied start-up clock (CLKUSR)* option in the Quartus II software
INIT_DONE	N/A if option is on. I/O if option is off.	Output open-drain	Status pin. Can be used to indicate when the device has initialized and is in user mode. When nCONFIG is low and during the beginning of configuration, the INIT_DONE pin is tri-stated and pulled high due to an external 1-kΩ pull-up. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low. When initialization is complete, the INIT_DONE pin will be released and pulled high and the FPGA enters user mode. Thus, the monitoring circuitry must be able to detect a low-to-high transition. This pin is enabled by turning on the *Enable INIT_DONE output* option in the Quartus II software.
DEV_OE	N/A if option is on. I/O if option is off.	Input	Optional pin that allows the user to override all tri-states on the device. When this pin is driven low, all I/O pins are tri-stated; when this pin is driven high, all I/O pins behave as programmed. This pin is enabled by turning on the *Enable device-wide output enable (DEV_OE)* option in the Quartus II software.
DEV_CLRn	N/A if option is on. I/O if option is off.	Input	Optional pin that allows you to override all clears on all device registers. When this pin is driven low, all registers are cleared; when this pin is driven high, all registers behave as programmed. This pin is enabled by turning on the *Enable device-wide reset (DEV_CLRn)* option in the Quartus II software.

JTAG pins must be kept stable before and during configuration. JTAG pin stability prevents accidental loading of JTAG instructions. Table 8–22 describes the dedicated JTAG pins.

Table 8–22. Dedicated JTAG Pins

Pin Name	User Mode	Pin Type	Description
TDI	N/A	Input	Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}.
TDO	N/A	Output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by leaving this pin unconnected.
TMS	N/A	Input	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}.
TCK	N/A	Input	The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND.
TRST	N/A	Input	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to IEEE Std. 1149.1. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND.

Volume 2: Configuration Handbook

101 Innovation Drive
San Jose, CA 95134
www.altera.com

CF5V2-4.0

Document last updated for Altera Complete Design Suite version: 11.0
Document publication date: June 2011

© 2011 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

QUALITY
ISO 9001:2008
NSAI Certified

Contents

Additional Information

How to Contact Altera Info–1
Typographic Conventions Info–1

Section I. FPGA Configuration Devices

Revision History I–1

Chapter 1. Enhanced Configuration Devices (EPC4, EPC8, and EPC16) Data Sheet

Features 1–1
Functional Description 1–2
FPGA Configuration 1–7
Configuration Signals 1–9
Fast Passive Parallel Configuration 1–9
Passive Serial Configuration 1–11
Concurrent Configuration 1–11
External Flash Interface 1–13
Intel-Flash-Based EPC Device Protection 1–15
Dynamic Configuration (Page Mode) 1–17
Real-Time Decompression 1–18
Programmable Configuration Clock 1–19
Flash In-System Programming (ISP) 1–20
JTAG-based Programming 1–20
Programming via External Flash Interface 1–21
Pin Description 1–22
Power-On Reset 1–25
Power Sequencing 1–25
Programming and Configuration File Support 1–26
IEEE Std. 1149.1 (JTAG) Boundary-Scan 1–28
Timing Information 1–29
Operating Conditions 1–31
Package 1–32
Package Layout Recommendation 1–34
Device Pin-Outs 1–35
Document Revision History 1–35

Chapter 2. Altera Enhanced Configuration Devices

Concurrent Configuration 2–1
Supported Schemes and Guidelines 2–1
Concurrent Configuration Using n-Bit PS Modes 2–2
Design Guidelines 2–5
Dynamic Configuration (Page Mode) Implementation Overview 2–5
Software Implementation (Convert Programming Files) 2–7
External Flash Memory Interface 2–9
Flash Memory Map 2–10
Logical Divisions 2–10
Physical Divisions 2–11
Interface Availability and Connections 2–11
Quartus II Software Support 2–12

Data Compression 2–14
Calculating the Size of Configuration Space 2–14
Clock Divider 2–15
Settings and Guidelines 2–15
Software Implementations 2–16
Conclusion 2–16
Document Revision History 2–16

Chapter 3. Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet

Features 3–1
Functional Description 3–2
Accessing Memory in Serial Configuration Devices 3–3
Active Serial FPGA Configuration 3–3
Serial Configuration Device Memory Access 3–7
Memory Array Organization 3–7
Operation Codes 3–15
Write Enable Operation 3–16
Write Disable Operation 3–17
Read Status Operation 3–17
Write Status Operation 3–20
Read Bytes Operation 3–21
Fast Read Operation 3–21
Read Silicon ID Operation 3–22
Read Device Identification Operation 3–23
Write Bytes Operation 3–24
Erase Bulk Operation 3–25
Erase Sector Operation 3–26
Power and Operation 3–26
Power Mode 3–26
Power-On Reset 3–27
Error Detection 3–27
Timing Information 3–27
Programming and Configuration File Support 3–30
Operating Conditions 3–31
Pin Information 3–32
Package 3–34
Ordering Code 3–34
Document Revision History 3–35

Chapter 4. Configuration Devices for SRAM-Based LUT Devices Data Sheet

Features 4–1
Functional Description 4–2
Device Configuration 4–6
Power and Operation 4–9
Power-On Reset (POR) 4–9
Error Detection Circuitry 4–9
3.3-V or 5.0-V Operation 4–10
Programming and Configuration File Support 4–11
IEEE Std. 1149.1 (JTAG) Boundary-Scan Testing 4–13
Timing Information 4–15
Operating Conditions 4–19
Pin Information 4–21
Package 4–24

Ordering Codes 4–25
Document Revision History 4–25

Chapter 5. Quad-Serial Configuration (EPCQ) Devices Datasheet
Supported Devices 5–1
Features 5–1
Memory Array Organization 5–2
Address Range for EPCQ128 5–2
Address Range for EPCQ256 5–4
Memory Operations 5–7
Timing Requirements 5–7
Addressing Mode 5–7
Summary of Operation Codes 5–8
4BYTEADDREN AND 4BYTEADDREX Operations 5–9
Write Enable Operation 5–9
Write Disable Operation 5–10
Read Status Operation 5–11
Write Status Operation 5–14
Read Bytes Operation 5–15
Fast Read Operation 5–16
Extended Dual Input Fast Read Operation 5–17
Extended Quad Input Fast Read Operation 5–18
Read Device Identification Operation 5–19
Write Bytes Operation 5–20
Extended Dual Input Fast Write Bytes Operation 5–21
Extended Quad Input Fast Write Bytes Operation 5–22
Erase Bulk Operation 5–23
Erase Sector Operation 5–24
Power Mode 5–24
Timing Information 5–25
Programming and Configuration File Support 5–26
Operating Conditions 5–27
Pin Information 5–29
Device Package and Ordering Code 5–32
Package 5–32
Ordering Code 5–32
Document Revision History 5–32

Section II. Software Settings
Revision History II–1

Chapter 6. Device Configuration Options
Configuration Scheme 6–1
Document Revision History 6–7

Chapter 7. Configuration File Formats
Generating Configuration Files 7–1
SRAM Object File (.sof) 7–3
Programmer Object File (.pof) 7–4
Raw Binary File (.rbf) 7–4
Raw Programming Data File (.rpd) 7–4
Hexadecimal (Intel-Format) File (.hex) or (.hexout) 7–4

Tabular Text File (.ttf) 7–5
Serial Bitstream File (.sbf) 7–5
Jam File (.jam) 7–5
Jam Byte-Code File (.jbc) 7–5
Document Revision History 7–6

Section III. Advanced Configuration Schemes

Revision History III–1

Chapter 8. Combining Different Configuration Schemes

Passive Serial and JTAG 8–2
Active Serial and JTAG 8–6
Document Revision History 8–7

This chapter provides additional information about the document and Altera.

This handbook provides comprehensive information about configuring Altera® FPGAs and configuration devices.

How to Contact Altera

To locate the most up-to-date information about Altera products, refer to the following table.

Contact *(1)*	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Product literature	Website	www.altera.com/literature
Non-technical support (General)	Email	nacomp@altera.com
(Software Licensing)	Email	authorization@altera.com

Note to Table:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

The following table shows the typographic conventions this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicate command names, dialog box titles, dialog box options, and other GUI labels. For example, **Save As** dialog box. For GUI elements, capitalization matches the GUI.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, software utility names, and GUI labels. For example, **\qdesigns** directory, **D:** drive, and **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Indicate document titles. For example, *Stratix IV Design Guidelines*.
italic type	Indicates variables. For example, *n* + 1. Variable names are enclosed in angle brackets (< >). For example, *<file name>* and *<project name>*.**pof** file.
Initial Capital Letters	Indicate keyboard keys and menu names. For example, the Delete key and the Options menu.
"Subheading Title"	Quotation marks indicate references to sections within a document and titles of Quartus II Help topics. For example, "Typographic Conventions."

Visual Cue	Meaning
Courier type	Indicates signal, port, register, bit, block, and primitive names. For example, `data1`, `tdi`, and `input`. The suffix `n` denotes an active-low signal. For example, `resetn`. Indicates command line commands and anything that must be typed exactly as it appears. For example, `c:\qdesigns\tutorial\chiptrip.gdf`. Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword `SUBDESIGN`), and logic function names (for example, `TRI`).
↵	An angled arrow instructs you to press the Enter key.
1., 2., 3., and a., b., c., and so on	Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ■ ■	Bullets indicate a list of items when the sequence of the items is not important.
	The hand points to information that requires special attention.
	A question mark directs you to a software help system with related information.
	The feet direct you to another document or website with related information.
	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
	A warning calls attention to a condition or possible situation that can cause you injury.
	The envelope links to the Email Subscription Management Center page of the Altera website, where you can sign up to receive update notifications for Altera documents.

Section I. FPGA Configuration Devices

This section provides information about Altera® configuration devices. The following chapters contain information about how to use these devices, feature descriptions, device pin tables, and package diagrams.

This section includes the following chapters:

- Chapter 1, Enhanced Configuration Devices (EPC4, EPC8, and EPC16) Data Sheet
- Chapter 2, Altera Enhanced Configuration Devices
- Chapter 3, Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet
- Chapter 4, Configuration Devices for SRAM-Based LUT Devices Data Sheet
- Chapter 5, Quad-Serial Configuration (EPCQ) Devices Datasheet

Revision History

Refer to each chapter for its own specific revision history. For information about when each chapter was updated, refer to the Chapter Revision Dates section, which appears in the complete handbook.

1. Enhanced Configuration Devices (EPC4, EPC8, and EPC16) Data Sheet

CF52002-2.9

Features

This chapter describes the EPC4, EPC8, and EPC16 enhanced configuration devices (EPC).

- Single-chip configuration solution for Altera® ACEX® 1K, APEX™ 20K (including APEX 20K, APEX 20KC, and APEX 20KE), APEX II, Arria® GX, Cyclone®, Cyclone II, FLEX® 10K (including FLEX 10KE and FLEX 10KA), Mercury™, Stratix® II, and Stratix II GX devices
- Contains 4-, 8-, and 16-Mbit flash memories for configuration data storage
  - On-chip decompression feature almost doubles the effective configuration density
- Standard flash die and a controller die combined into single stacked chip package
- External flash interface supports parallel programming of flash and external processor access to unused portions of memory
  - Flash memory block/sector protection capability via external flash interface
  - Supported in EPC16 and EPC4 devices
- Page mode support for remote and local reconfiguration with up to eight configurations for the entire system
  - Compatible with Stratix series Remote System Configuration feature
- Supports byte-wide configuration mode fast passive parallel (FPP); 8-bit data output per DCLK cycle
- Supports true *n*-bit concurrent configuration (*n* = 1, 2, 4, and 8) of Altera FPGAs
- Pin-selectable 2-ms or 100-ms power-on reset (POR) time
- Configuration clock supports programmable input source and frequency synthesis
  - Multiple configuration clock sources supported (internal oscillator and external clock input pin)
  - External clock source with frequencies up to 100 MHz
  - Internal oscillator defaults to 10 MHz; Programmable for higher frequencies of 33, 50, and 66 MHz
  - Clock synthesis supported via user programmable divide counter
- Available in the 100-pin plastic quad flat pack (PQFP) and the 88-pin Ultra FineLine BGA® (UFBGA) packages
  - Vertical migration between all devices supported in the 100-pin PQFP package
- Supply voltage of 3.3 V (core and I/O)

© 2011 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

- Hardware compliant with IEEE Std. 1532 in-system programmability (ISP) specification
- Supports ISP via Jam Standard Test and Programming Language (STAPL)
- Supports JTAG boundary scan
- *n*INIT_CONF pin allows private JTAG instruction to start FPGA configuration
- Internal pull-up resistor on *n*INIT_CONF always enabled
- User programmable weak internal pull-up resistors on *n*CS and OE pins
- Internal weak pull-up resistors on external flash interface address and control lines, bus hold on data lines
- Standby mode with reduced power consumption

For more information about FPGA configuration schemes and advanced features, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.

Functional Description

The Altera enhanced configuration device is a single-device, high-speed, advanced configuration solution for very high-density FPGAs. The core of an enhanced configuration device is divided into two major blocks: a configuration controller and a flash memory. The flash memory is used to store configuration data for systems made up of one or more Altera FPGAs. Unused portions of the flash memory can be used to store processor code or data that can be accessed via the external flash interface after FPGA configuration is complete. Table 1–1 summarizes the features of Altera configuration devices and the amount of configuration space they hold.

Table 1–1. Altera Configuration Devices

Device	Memory Size (bits)	On-Chip Decompression Support	ISP Support	Cascading Support	Reprogrammable	Operating Voltage (V)
EPC4	4,194,304	Yes	Yes	No	Yes	3.3
EPC8	8,388,608	Yes	Yes	No	Yes	3.3
EPC16	16,777,216	Yes	Yes	No	Yes	3.3

Table 1–2 lists the supported configuration devices required to configure an ACEX 1K, APEX 1K, APEX 20K, APEX 20KC, APEX 20KE, APEX II, Arria GX, Cyclone, Cyclone II, FLEX 10K, FLEX 10KA, FLEX 10KE, Stratix, Stratix GX, Stratix II, Stratix II GX, or Mercury device.

Table 1–2. Configuration Devices Required (Part 1 of 3)

Family	Device	Data Size (Bits) *(1)*	EPC4 *(2)*	EPC8 *(2)*	EPC16 *(2)*
Arria GX	EP1AGX20C	9,640,672	—	—	1
	EP1AGX35C EP1AGX35D	9,640,672	—	—	1
	EP1AGX50C EP1AGX50D	16,951,824	—	—	1
	EP1AGX60C EP1AGX60D EP1AGX60E	16,951,824	—	—	1
	EP1AGX90E	25,699,104	—	—	1
Stratix	EP1S10	3,534,640	1	1	1
	EP1S20	5,904,832	1	1	1
	EP1S25	7,894,144	—	1	1
	EP1S30	10,379,368	—	1	1
	EP1S40	12,389,632	—	1	1
	EP1S60	17,543,968	—	—	1
	EP1S80	23,834,032	—	—	1
Stratix GX	EP1SGX10	3,534,640	1	1	1
	EP1SGX25	7,894,144	—	1	1
	EP1SGX40	12,389,632	—	1	1
Stratix II	EP2S15	4,721,544	1	1	1
	EP2S30	9,640,672	—	1	1
	EP2S60	16,951,824	—	—	1
	EP2S90	25,699,104	—	—	—
	EP2S130	37,325,760	—	—	—
	EP2S180	49,814,760	—	—	—
Stratix II GX	EP2SGX30C	9,640,672	—	—	1
	EP2SGX30D	9,640,672	—	—	1
	EP2SGX60C	16,951,824	—	—	1
	EP2SGX60D	16,951,824	—	—	1
	EP2SGX60E	16,951,824	—	—	1
	EP2SGX90E	25,699,104	—	—	—
	EP2SGX90F	25,699,104	—	—	—
	EP2SGX130G	37,325,760	—	—	—

Table 1–2. Configuration Devices Required (Part 2 of 3)

Family	Device	Data Size (Bits) *(1)*	EPC4 *(2)*	EPC8 *(2)*	EPC16 *(2)*
Cyclone	EP1C3	627,376	1	1	1
	EP1C4	924,512	1	1	1
	EP1C6	1,167,216	1	1	1
	EP1C12	2,326,528	1	1	1
	EP1C20	3,559,608	1	1	1
Cyclone II	EP2C5	1,223,980	1	1	1
	EP2C8	1,983,792	1	1	1
	EP2C20	3,930,986	1	1	1
	EP2C35	7,071,234	—	1	1
	EP2C50	9,122,148	—	1	1
	EP2C70	10,249,694	—	1	1
ACEX 1K	EP1K10	159,160	1	1	1
	EP1K30	473,720	1	1	1
	EP1K50	784,184	1	1	1
	EP1K100	1,335,720	1	1	1
APEX 20K	EP20K100	993,360	1	1	1
	EP20K200	1,950,800	1	1	1
	EP20K400	3,880,720	1	1	1
APEX 20KC	EP20K200C	196,8016	1	1	1
	EP20K400C	390,9776	1	1	1
	EP20K600C	567,3936	1	1	1
	EP20K1000C	8,960,016	—	1	1
APEX 20KE	EP20K30E	354,832	1	1	1
	EP20K60E	648,016	1	1	1
	EP20K100E	1,008,016	1	1	1
	EP20K160E	1,524,016	1	1	1
	EP20K200E	1,968,016	1	1	1
	EP20K300E	2,741,616	1	1	1
	EP20K400E	3,909,776	1	1	1
	EP20K600E	5,673,936	1	1	1
	EP20K1000E	8,960,016	—	1	1
	EP20K1500E	12,042,256	—	1	1
APEX II	EP2A15	4,358,512	1	1	1
	EP2A25	6,275,200	1	1	1
	EP2A40	9,640,528	—	1	1
	EP2A70	17,417,088	—	—	1

Table 1–2. Configuration Devices Required (Part 3 of 3)

Family	Device	Data Size (Bits) *(1)*	EPC4 *(2)*	EPC8 *(2)*	EPC16 *(2)*
FLEX 10K	EPF10K10	118,000	1	1	1
	EPF10K20	231,000	1	1	1
	EPF10K30	376,000	1	1	1
	EPF10K40	498,000	1	1	1
	EPF10K50	621,000	1	1	1
	EPF10K70	892,000	1	1	1
	EPF10K100	1,200,000	1	1	1
FLEX 10KA	EPF10K10A	120,000	1	1	1
	EPF10K30A	406,000	1	1	1
	EPF10K50V	621,000	1	1	1
	EPF10K100A	1,200,000	1	1	1
	EPF10K130V	1,600,000	1	1	1
	EPF10K250A	3,300,000	1	1	1
FLEX 10KE	EPF10K30E	473,720	1	1	1
	EPF10K50E	784,184	1	1	1
	EPF10K50S	784,184	1	1	1
	EPF10K100B	1,200,000	1	1	1
	EPF10K100E	1,335,720	1	1	1
	EPF10K130E	1,838,360	1	1	1
	EPF10K200E	2,756,296	1	1	1
	EPF10K200S	2,756,296	1	1	1
Mercury	EP1M120	1,303,120	1	—	1
	EP1M350	4,394,032	1	—	1

Notes to Table 1–2:

(1) Raw Binary Files (**.rbf**) were used to determine these sizes.

(2) These values with the enhanced configuration device compression feature enabled.

For more information about additional enhanced configuration devices, refer to the Process Change Notification *PCN0506: Addition of Intel Flash Memory As Source For EPC4, EPC8 and EPC16 Enhanced Configuration Devices* and *Using the Intel Flash Memory-Based EPC4, EPC8 and EPC16 Devices* white paper.

EPC devices support three different types of flash memory. Table 1–3 lists the supported flash memory for all EPC devices.

Table 1–3. Enhanced Configuration Devices Flash Memory (Part 1 of 2)

Device	Grade	Package	Flash Memory	
			Leaded	Lead-Fee
EPC4	Commercial	PQFP 100	Intel *(1)* or Micron	Intel *(1)* or Micron
	Industrial	PQFP 100	Intel *(1)* or Micron	Intel *(1)*

Table 1–3. Enhanced Configuration Devices Flash Memory (Part 2 of 2)

Device	Grade	Package	Flash Memory	
			Leaded	Lead-Fee
EPC8	Commercial/ Industrial	PQFP 100	Intel *(1)* or Sharp	Intel *(1)*
EPC16	Commercial	UBGA 88	Intel *(1)* or Sharp	Intel *(1)* or Sharp
	Industrial	UBGA 88	Intel *(1)* or Sharp	Intel *(1)*
	Military	UBGA 88	Intel *(1)*	Intel *(1)*
	Commercial/ Industrial	PQFP 100	Intel *(1)* or Sharp	Intel *(1)*

Note to Table 1–3:

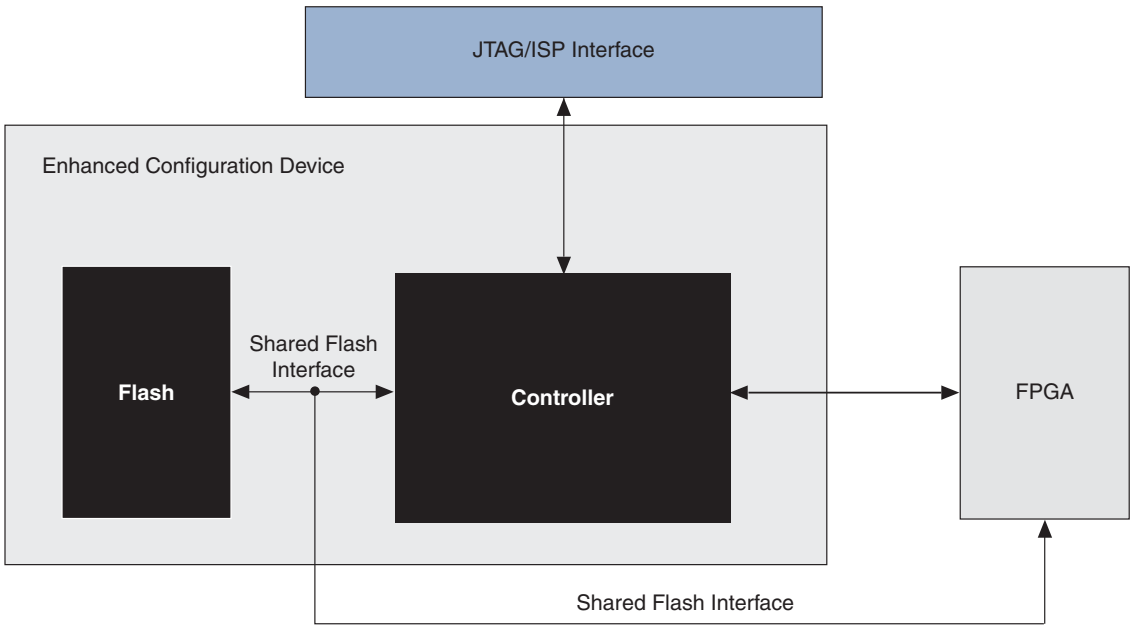
(1) For more information, refer to the Process Change Notification *PCN0506: Addition of Intel Flash Memory As Source for EPC4, EPC8 and EPC16 Enhanced Configuration Devices*.

The external flash interface is currently supported in the EPC16 and EPC4 devices. For information about using this feature in the EPC8 device, contact Altera Applications at **www.altera.com/support**.

Enhanced configuration devices have a 3.3-V core and I/O interface. The controller chip is a synchronous system that implements the various interfaces and features. Figure 1–1 shows a block diagram of the enhanced configuration device. The controller chip features three separate interfaces:

- A configuration interface between the controller and the Altera FPGAs
- A JTAG interface on the controller that enables ISP of the flash memory
- An external flash interface that the controller shares with an external processor, or FPGA implementing a Nios® embedded processor (interface available after ISP and configuration)

Figure 1–1. Enhanced Configuration Device Block Diagram



The enhanced configuration device features multiple configuration schemes. In addition to supporting the traditional passive serial (PS) configuration scheme for a single device or a serial device chain, the enhanced configuration device features concurrent configuration and parallel configuration. With the concurrent configuration scheme, up to eight PS device chains can be configured simultaneously. In the FPP configuration scheme, 8-bits of data are clocked into the FPGA each cycle. These schemes offer significantly reduced configuration times over traditional schemes.

Furthermore, the enhanced configuration device features a dynamic configuration or page mode feature. This feature allows you to dynamically reconfigure all the FPGAs in your system with new images stored in the configuration memory. Up to eight different system configurations or pages can be stored in memory and selected using the PGM[2..0] pins. Your system can be dynamically reconfigured by selecting one of the eight pages and initiating a reconfiguration cycle.

This page mode feature combined with the external flash interface allows remote and local updates of system configuration data. The enhanced configuration devices are compatible with the Stratix Remote System Configuration feature.

For more information about Stratix Remote System Configuration, refer to the *Remote System Configuration with Stratix & Stratix GX Devices* chapter in the *Stratix Device Handbook*.

Other user programmable features include:

- Real-time decompression of configuration data
- Programmable configuration clock (DCLK)
- Flash ISP
- Programmable power-on-reset delay (PORSEL)

FPGA Configuration

FPGA configuration is managed by the configuration controller chip. This process includes reading configuration data from the flash memory, decompressing it if necessary, transmitting configuration data via the appropriate DATA[] pins, and handling error conditions.

After POR, the controller determines the user-defined configuration options by reading its option bits from the flash memory. These options include the configuration scheme, configuration clock speed, decompression, and configuration page settings. The option bits are stored at flash address location 0x8000 (word address) and occupy 512-bits or 32-words of memory. These options bits are read using the internal flash interface and the default 10 MHz internal oscillator.

After obtaining the configuration settings, the configuration controller chip checks if the FPGA is ready to accept configuration data by monitoring the *n*STATUS and CONF_DONE lines. When the FPGA is ready (*n*STATUS is high and CONF_DONE is low), the controller begins data transfer using the DCLK and DATA[] output pins. The controller selects the configuration page to be transmitted to the FPGA by sampling its PGM[2..0] pins after POR or reset.

The function of the configuration unit is to transmit decompressed data to the FPGA, depending on the configuration scheme. The enhanced configuration device supports four concurrent configuration modes, with *n* = 1, 2, 4, or 8 (where *n* is the number of bits that are sent per DCLK cycle on the DATA[*n*] lines). The value *n* = 1 corresponds to the traditional PS configuration scheme. The values *n* = 2, 4, and 8 correspond to concurrent configuration of 2, 4, or 8 different PS configuration chains, respectively. Additionally, the FPGA can be configured in FPP mode, where eight bits of DATA are clocked into the FPGA per DCLK cycle. Depending on the configuration bus width (*n*), the circuit shifts uncompressed configuration data to the valid DATA[*n*] pins. Unused DATA[] pins drive low.

In addition to transmitting configuration data to the FPGAs, the configuration circuit is also responsible for pausing configuration whenever there is insufficient data available for transmission. This occurs when the flash read bandwidth is lower than the configuration write bandwidth. Configuration is paused by stopping the DCLK to the FPGA, when waiting for data to be read from the flash or for data to be decompressed. This technique is called "Pausing DCLK".

The enhanced configuration device flash-memories feature a 90-ns access time (approximately 10 MHz). Hence, the flash read bandwidth is limited to about 160 megabits per second (Mbps) (16-bit flash data bus, DQ[], at 10 MHz). However, the configuration speeds supported by Altera FPGAs are much higher and translate to high configuration write bandwidths. For instance, 100-MHz Stratix FPP configuration requires data at the rate of 800 Mbps (8-bit DATA[] bus at 100 MHz). This is much higher than the 160 Mbps the flash memory can support, and is the limiting factor for configuration time. Compression increases the effective flash-read bandwidth as the same amount of configuration data takes up less space in the flash memory after compression. Since Stratix configuration data compression ratios are approximately two, the effective read bandwidth doubles to about 320 Mbps.

Finally, the configuration controller also manages errors during configuration. A CONF_DONE error occurs when the FPGA does not de-assert its CONF_DONE signal within 64 DCLK cycles after the last bit of configuration data is transmitted. When a CONF_DONE error is detected, the controller pulses the OE line low, which pulls *n*STATUS low and triggers another configuration cycle.

A cyclical redundancy check (CRC) error occurs when the FPGA detects corruption in the configuration data. This corruption could be a result of noise coupling on the board such as poor signal integrity on the configuration signals. When this error is signaled by the FPGA (by driving the *n*STATUS line low), the controller stops configuration. If the Auto-Restart Configuration After Error option is enabled in the FPGA, it releases its *n*STATUS signal after a reset time-out period and the controller attempts to reconfigure the FPGA.

After the FPGA configuration process is complete, the controller drives DCLK low and the DATA[] pins high. Additionally, the controller tri-states its internal interface to the flash memory, enables the weak internal pull-ups on the flash address and control lines, and enables bus-keep circuits on flash data lines.

The following sections briefly describe the different configuration schemes supported by the enhanced configuration device: FPP, PS, and concurrent configuration.

For detailed information about using these schemes to configure your Altera FPGA, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.

Configuration Signals

Table 1–4 lists the configuration signal connections between the enhanced configuration device and Altera FPGAs.

Table 1–4. Configuration Signals

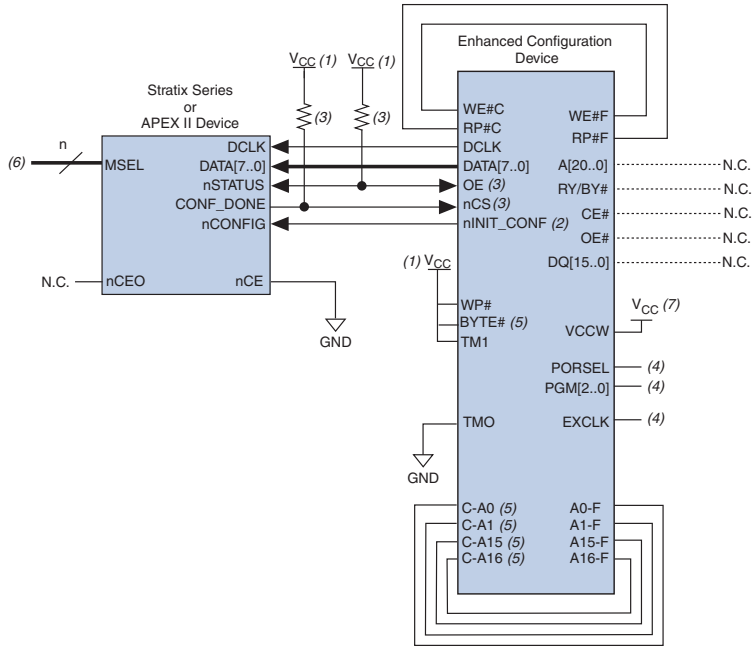
Enhanced Configuration Device Pin	Altera FPGA Pin	Description
DATA[]	DATA[]	Configuration data transmitted from the configuration device to the FPGA, which is latched on the rising edge of DCLK.
DCLK	DCLK	Configuration device generated clock used by the FPGA to latch configuration data provided on the DATA[] pins.
*n*INIT_CONF, which	*n*CONFIG	Open-drain output from the configuration device that is used to start FPGA reconfiguration using the initiate configuration (INIT_CONF) JTAG instruction. This connection is not needed if the INIT_CONF JTAG instruction is not needed. If *n*INIT_CONF is not connected to *n*CONFIG, *n*CONFIG must be tied to V_{CC} either directly or through a pull-up resistor.
OE	*n*STATUS	Open-drain bidirectional configuration status signal, which is driven low by either device during POR and to signal an error during configuration. Low pulse on OE resets the enhanced configuration device controller.
*n*CS	CONF_DONE	Configuration done output signal driven by the FPGA.

Fast Passive Parallel Configuration

Stratix series and APEX II devices can be configured using the enhanced configuration device in FPP mode. In this mode, the enhanced configuration device sends a byte of data on the DATA[7..0] pins, which connect to the DATA[7..0] input pins of the FPGA, per DCLK cycle. Stratix series and APEX II FPGAs receive byte-wide configuration data per DCLK cycle. Figure 1–2 shows the enhanced configuration

device in FPP configuration mode. In this figure, the external flash interface is not used and hence most flash pins are left unconnected (with the few noted exceptions). For specific details about configuration interface connections including pull-up resistor values, supply voltages, and MSEL pin settings, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.

Figure 1–2. FPP Configuration



Notes to Figure 1–2:

(1) The V_{CC} should be connected to the same supply voltage as the configuration device.

(2) The *n*INIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active. This means an external pull-up resistor is not required on the *n*INIT_CONF / *n*CONFIG line. The *n*INIT_CONF pin does not need to be connected if its functionality is not used. If *n*INIT_CONF is not used, *n*CONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' OE and *n*CS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus® II software. To turn off the internal pull-up resistors, check the **Disable n**CS and OE pull-ups on configuration device option when generating programming files.

(4) For PORSEL, PGM[], and EXCLK pin connections, refer to Table 1–10.

(5) In the 100-pin PQFP package, you must externally connect the following pins: C-A0 to F-A0, C-A1 to F-A1, C-A15 to F-A15, C-A16 to F-A16, and BYTE# to V_{CC}. Additionally, you must make the following pin connections in both 100-pin PQFP and 88-pin UFBGA packages: C-RP# to F-RP#, C-WE# to F-WE#, TM1 to V_{CC}, TM0 to GND, and WP# to V_{CC}.

(6) Connect the FPGA MSEL[] input pins to select the FPP configuration mode. For details, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.

(7) To protect Intel Flash based EPC devices content, isolate the V_{CCW} supply from V_{CC}. For more information, refer section "Intel-Flash-Based EPC Device Protection" on page 1–15.

Multiple FPGAs can be configured using a single enhanced configuration device in FPP mode. In this mode, multiple Stratix series FPGAs, APEX II FPGAs, or both, are cascaded together in a daisy chain.

After the first FPGA completes configuration, its *n*CEO pin asserts to activate the *n*CE pin for the second FPGA, which prompts the second device to start capturing configuration data. In this setup, the FPGAs CONF_DONE pins are tied together, and hence all devices initialize and enter user mode simultaneously. If the enhanced configuration device or one of the FPGAs detects an error, configuration stops (and simultaneously restarts) for the whole chain because the *n*STATUS pins are tied together.

While Altera FPGAs can be cascaded in a configuration chain, the enhanced configuration devices cannot be cascaded to configure larger devices or chains.

For configuration schematics and more information about multi-device FPP configuration, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.

Passive Serial Configuration

APEX 20KC, APEX 20KE, APEX 20K, APEX II, Cyclone series, FLEX 10K, and Stratix series devices can be configured using enhanced configuration devices in the PS mode. This mode is similar to the FPP mode, with the exception that only one bit of data (DATA[0]) is transmitted to the FPGA per DCLK cycle. The remaining DATA[7..1] output pins are unused in this mode and drive low.

The configuration schematic for PS configuration of a single FPGA or single serial chain is identical to the FPP schematic (with the exception that only DATA[0] output from the enhanced configuration device connects to the FPGA DATA0 input pin; remaining DATA[7..1] pins are left floating).

For configuration schematics and more information about multi-device PS configuration, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.

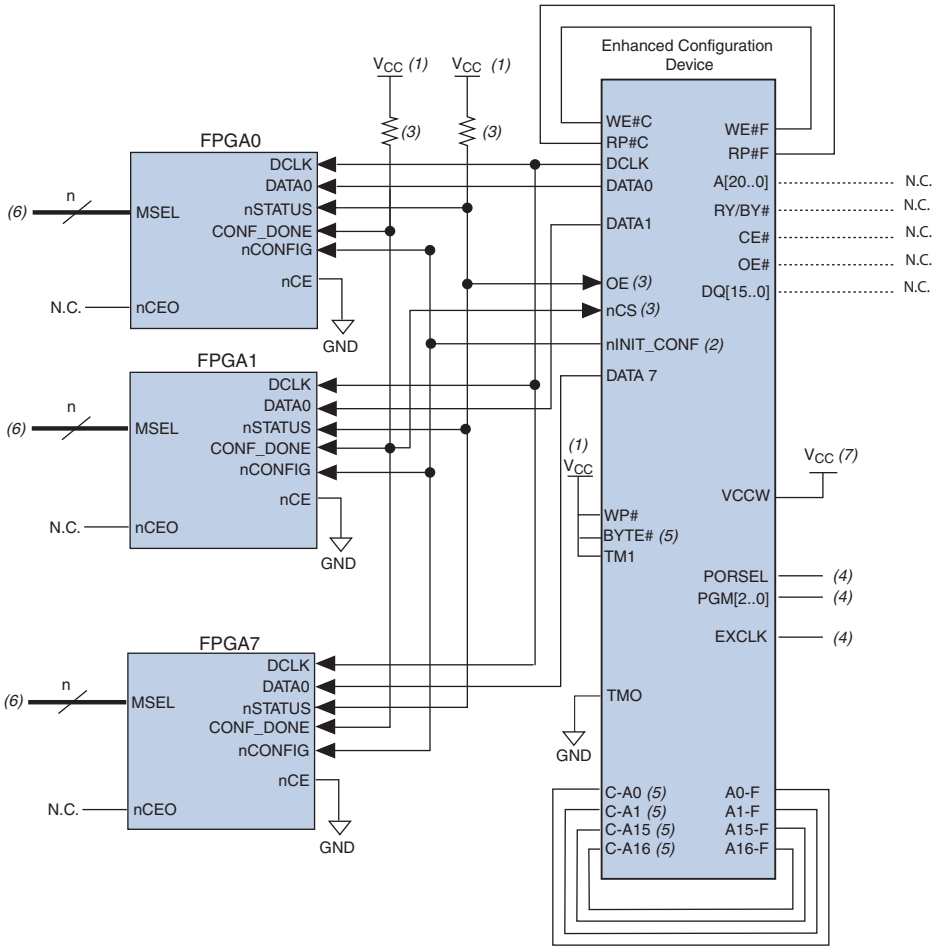
Concurrent Configuration

Enhanced configuration devices support concurrent configuration of multiple FPGAs (or FPGA chains) in PS mode. Concurrent configuration is when the enhanced configuration device simultaneously outputs *n* bits of configuration data on the DATA[*n*-1..0] pins (*n* = 1, 2, 4, or 8), and each DATA[] line serially configures a different FPGA (chain). The number of concurrent serial chains is user-defined via the Quartus II software and can be any number from 1 to 8. For example, three concurrent chains you can select the 4-bit PS mode, and connect the least significant DATA bits to the FPGAs or FPGA chains. Leave the most significant DATA bit (DATA[3]) unconnected. Similarly, for 5-, 6-, or 7-bit concurrent chains you can select the 8-bit PS mode.

Figure 1–3 shows the schematic for configuring multiple FPGAs concurrently in the PS mode using an enhanced configuration device.

For specific details about configuration interface connections including pull-up resistor values, supply voltages, and MSEL pin settings, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.

Figure 1–3. Concurrent Configuration of Multiple FPGAs in PS Mode (n = 8)



Notes to Figure 1–3:

(1) Connect V_{CC} to the same supply voltage as the configuration device.

(2) The *n*INIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active. This means an external pull-up resistor is not required on the *n*INIT_CONF/*n*CONFIG line. The *n*INIT_CONF pin does not need to be connected if its functionality is not used. If *n*INIT_CONF is not used, *n*CONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) The enhanced configuration devices' OE and *n*CS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

(4) For PORSEL, PGM[], and EXCLK pin connections, refer to Table 1–10.

(5) In the 100-pin PQFP package, you must externally connect the following pins: C-A0 to F-A0, C-A1 to F-A1, C-A15 to F-A15, C-A16 to F-A16, and BYTE# to V_{CC}. Additionally, you must make the following pin connections in both 100-pin PQFP and 88-pin UFBGA packages: C-RP# to F-RP#, C-WE# to F-WE#, TM1 to V_{CC}, TM0 to GND, and WP# to V_{CC}.

(6) Connect the FPGA MSEL[] input pins to select the PS configuration mode. For details, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.

(7) To protect Intel Flash based EPC devices content, isolate the V_{CCW} supply from V_{CC}. For more information, refer section "Intel-Flash-Based EPC Device Protection" on page 1–15.

Table 1–5 summarizes the concurrent PS configuration modes supported in the enhanced configuration device.

Table 1–5. Enhanced Configuration Devices in PS Mode

Mode Name	Mode (*n* =) *(1)*	Used Outputs	Unused Outputs
Passive serial mode	1	DATA0	DATA[7..1] drive low
Multi-device passive serial mode	2	DATA[1..0]	DATA[7..2] drive low
Multi-device passive serial mode	4	DATA[3..0]	DATA[7..4] drive low
Multi-device passive serial mode	8	DATA[7..0]	—

Note to Table 1–5:

(1) This is the number of valid DATA outputs for each configuration mode.

For configuration schematics and more information about concurrent configurations, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.

External Flash Interface

The enhanced configuration devices support external FPGA or processor access to its flash memory. The unused portions of the flash memory can be used by the external device to store code or data. This interface can also be used in systems that implement remote configuration capabilities. Configuration data within a particular configuration page can be updated via the external flash interface and the system could be reconfigured with the new FPGA image. This interface is also useful to store Nios boot code, application code, or both.

For more information about the Stratix remote configuration feature, refer to the *Remote System Configuration with Stratix & Stratix GX Devices* chapter in the *Stratix Device Handbook*.

The address, data, and control ports of the flash memory are internally connected to the enhanced configuration device controller and to external device pins. An external source can drive these external device pins to access the flash memory when the flash interface is available.

This external flash interface is a shared bus interface with the configuration controller chip. The configuration controller is the primary bus master. Since there is no bus arbitration support, the external device can only access the flash interface when the controller has tri-stated its internal interface to the flash. Simultaneous access by the controller and the external device will cause contention, and result in configuration and programming failures.

Since the internal flash interface is directly connected to the external flash interface pins, controller flash access cycles will toggle the external flash interface pins. The external device must be able to tri-state its flash interface during these operations and ignore transitions on the flash interface pins.

The external flash interface signals cannot be shared between multiple enhanced configuration devices because this causes contention during in-system programming and configuration. During these operations, the controller chips inside the enhanced configuration devices are actively accessing flash memory. Therefore, enhanced configuration devices do not support shared flash bus interfaces.

The enhanced configuration device controller chip accesses flash memory during:

- FPGA configuration—reading configuration data from flash
- JTAG-based flash programming—storing configuration data in flash
- At POR—reading option bits from flash

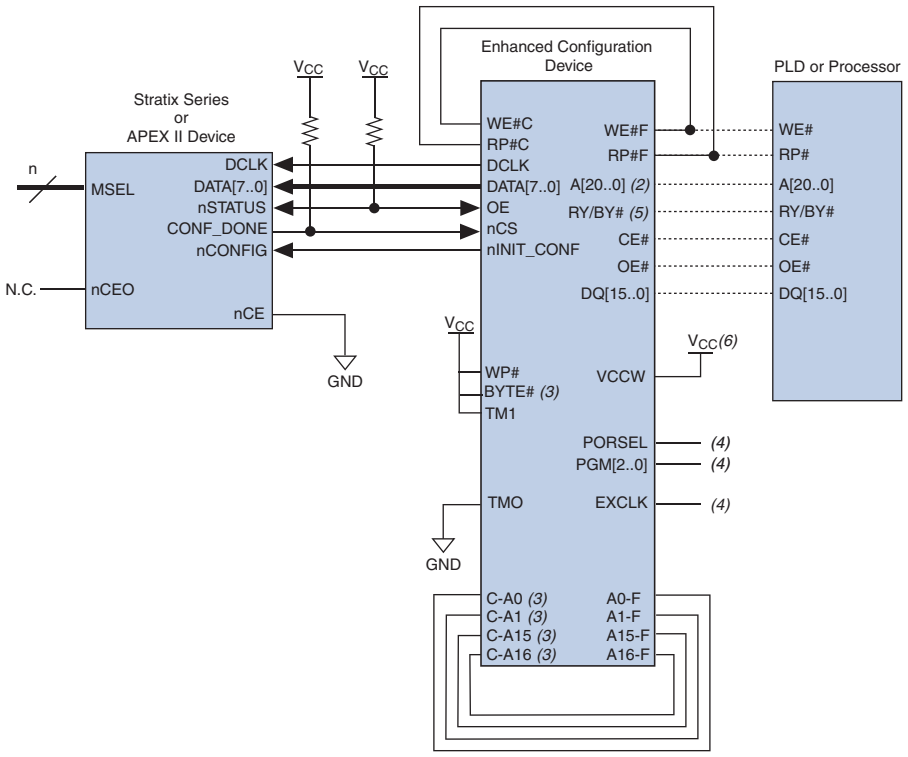
During these operations, the external FPGA or processor must tri-state its interface to the flash memory. After configuration and programming, the enhanced configuration device's controller tri-states the internal interface and goes into an idle mode. To interrupt a configuration cycle in order to access the flash via the external flash interface, the external device can hold the FPGA's *n*CONFIG input low. This keeps the configuration device in reset by holding the *n*STATUS-OE line low, allowing external flash access.

For more information about the software support for the external flash interface feature, refer to the *Altera Enhanced Configuration Devices* chapter in volume 2 of the *Configuration Handbook*. For details about flash commands, timing, memory organization, and write protection features, refer to the following documents:

- For Micron flash-based EPC4, refer to the *Micron Flash Memory MT28F400B3 Data Sheet* at www.micron.com.
- For Sharp flash-based EPC16, refer to the *Sharp LHF16J06 Data Sheet Flash Memory Used in EPC16 Devices* at www.sharpsma.com.
- For the *Intel Advanced Boot Block Flash Memory (B3) 28F008/800B3, 28F016/160B3, 28F320B3, 28F640B3 Datasheet*, visit www.intel.com.

Figure 1–4 shows an FPP configuration schematic with the external flash interface in use.

Figure 1–4. FPP Configuration with External Flash Interface *(Note 1)*



Notes to Figure 1–4:

(1) For external flash interface support in EPC8 enhanced configuration device, contact Altera Applications.

(2) Pin A20 in EPC16 devices, pins A20 and A19 in EPC8 devices, and pins A20, A19, and A18 in EPC4 devices should be left floating. These pins should not be connected to any signal; they are no-connect pins.

(3) In the 100-pin PQFP package, you must externally connect the following pins: C-A0 to F-A0, C-A1 to F-A1, C-A15 to F-A15, C-A16 to F-A16, and BYTE# to V_{CC}. Additionally, you must make the following pin connections in both 100-pin PQFP and 88-pin UFBGA packages: C-RP# to F-RP#, C-WE# to F-WE#, TM1 to V_{CC}, TM0 to GND, and WP# to V_{CC}.

(4) For PORSEL, PGM[], and EXCLK pin connections, refer to Table 1–10.

(5) RY/BY# pin is only available for Sharp flash-based EPC8 and EPC16.

(6) To protect Intel Flash based EPC devices content, isolate the V_{CCW} supply from V_{CC}. For more information, refer section "Intel-Flash-Based EPC Device Protection" on page 1–15.

Intel-Flash-Based EPC Device Protection

In the absence of the lock bit protection feature in the EPC4, EPC8, and EPC16 devices with Intel flash, Altera recommends four methods to protect the Intel Flash content in EPC4, EPC8, and EPC16 devices. Any method alone is sufficient to protect the flash. The methods are listed here in the order of descending protection level:

1. Using an RP# of less than 0.3 V on power-up and power-down for a minimum of 100 ns to a maximum 25 ms disables all control pins, making it impossible for a write to occur.

2. Using $V_{PP} < V_{PPLK}$, where the maximum value of V_{PPLK} is 1 V, disables writes. $V_{PP} < V_{PPLK}$ means programming or writes cannot occur. V_{PP} is a programming supply voltage input pin on the Intel flash. V_{PP} is equivalent to the VCCW pin on EPC devices.
3. Using a high CE# disables the chip. The requirement for a write is a low CE# and low WE#. A high CE# by itself prevents writes from occurring.
4. Using a high WE# prevent writes because a write only occurs when the WE# is low.

Performing all four methods simultaneously is the safest protection for the flash content.

The ideal power-up sequence is as follows:

1. Power-up V_{CC}.
2. Maintain $V_{PP} < V_{PPLK}$ until V_{CC} is fully powered up.
3. Power-up V_{PP}.
4. Drive RP# low during the entire power-up process. RP# must be released high within 25 ms after V_{PP} is powered up.

☞ CE# and WE# must be high for the entire power-up sequence.

The ideal power-down sequence is as follows:

1. Drive RP# low for 100ns before power-down.
2. Power-down $V_{PP} < V_{PPLK}$.
3. Power-down V_{CC}.
4. Drive RP# low during the entire power-down process.

☞ CE# and WE# must be high for the entire power-down sequence.

The RP# pin is not internally connected to the controller. Therefore, an external loop-back connection between C-RP# and F-RP# must be made on the board even when you are not using the external device to the RP# pin with the loop-back. Always tri-state RP# when the flash is not in use.

If an external power-up monitoring circuit is connected to the RP# pin with the loop-back, use the following guidelines to avoid contention on the RP# line:

- The power-up sequence on the 3.3-V supply should complete within 50 ms of power-up. The 3.3-V V_{CC} should reach the minimum V_{CC} before 50 ms and RP# should then be released.
- RP# should be driven low by the power-up monitoring circuit during power-up. After power-up, RP# should be tri-stated externally by the power-up monitoring circuit.

If the preceding guidelines cannot be completed within 50 ms, then the OE pin must be driven low externally until RP# is ready to be released.

Dynamic Configuration (Page Mode)

The dynamic configuration (or page mode) feature allows the enhanced configuration device to store up to eight different sets of designs for all the FPGAs in your system. You can then choose which page (set of configuration files) the enhanced configuration device should use for FPGA configuration.

Dynamic configuration or the page mode feature enables you to store a minimum of two pages: a factory default or fail-safe configuration, and an application configuration. The fail-safe configuration page could be programmed during system production, while the application configuration page could support remote or local updates. These remote updates could add or enhance system features and performance. However, with remote update capabilities comes the risk of possible corruption of configuration data. In the event of such a corruption, the system could automatically switch to the fail-safe configuration and avoid system downtime.

The enhanced configuration device page mode feature works with the Stratix Remote System Configuration feature, to enable intelligent remote updates to your systems.

For more information about remotely updating Stratix FPGAs, refer to *Remote System Configuration with Stratix & Stratix GX Devices* in the *Stratix Device Handbook*.

The three PGM[2..0] input pins control which page is used for configuration, and these pins are sampled at the start of each configuration cycle when OE goes high. The page mode selection allows you to dynamically reconfigure the functionality of your FPGA by switching the PGM[2..0] pins and asserting nCONFIG. Page 0 is defined as the default page and the PGM[2] pin is the most significant bit (MSB).

The PGM[2..0] input pins must not be left floating on your board, regardless of whether this feature is used or not. When this feature is not used, connect the PGM[2..0] pins to GND to select the default page 000.

The enhanced configuration device pages are dynamically sized regions in memory. The start address and length of each page is programmed into the option-bit space of the flash memory during initial programming. All subsequent configuration cycles will sample the PGM[] pins and use the option-bit information to jump to the start of the corresponding configuration page. Each page must have configuration files for all FPGAs in your system that are connected to that enhanced configuration device.

For example, if your system requires three configuration pages and includes two FPGAs, each page will store two SRAM Object Files (**.sof**) for a total of six **.sof** in the configuration device.

Furthermore, all enhanced configuration device configuration schemes (PS, FPP, and concurrent PS) are supported with the page-mode feature. The number of pages, devices, or both, that can be configured using a single enhanced configuration device is only limited by the size of the flash memory.

For detailed information about the page-mode feature implementation and programming file generation steps using the Quartus II software, refer to the *Altera Enhanced Configuration Devices* chapter in volume 2 of the *Configuration Handbook*.

Real-Time Decompression

Enhanced configuration devices support on-chip real time decompression of configuration data. FPGA configuration data is compressed by the Quartus II software and stored in the enhanced configuration device. During configuration, the decompression engine inside the enhanced configuration device will decompress or expand configuration data. This feature increases the effective-configuration density of the enhanced configuration device up to 7, 15, or 30 Mbits in the EPC4, EPC8, and EPC16, respectively.

The enhanced configuration device also supports a parallel 8-bit data bus to the FPGA to reduce configuration time. However, in some cases, the FPGA data-transfer time is limited by the flash-read bandwidth. For example, when configuring an APEX II device in FPP (byte-wide data per cycle) mode at a configuration speed of 66 MHz, the FPGA write bandwidth is equal to 8 bits × 66 MHz = 528 Mbps. The flash read interface, however, is limited to approximately 10 MHz (since the flash access time is ~90 ns). This translates to a flash-read bandwidth of 16 bits × 10 MHz = 160 Mbps. Hence, the configuration time is limited by the flash-read time.

When configuration data is compressed, the amount of data that needs to be read out of the flash is reduced by about 50%. If 16 bits of compressed data yields 30 bits of uncompressed data, the flash-read bandwidth increases to 30 bits × 10 MHz = 300 Mbps, reducing overall configuration time.

You can enable the controller's decompression feature in the Quartus II software, **Configuration Device Options** window by turning on **Compression Mode**.

The decompression feature supported in the enhanced configuration devices is different from the decompression feature supported by the Stratix II FPGAs and the Cyclone series. When configuring Stratix II FPGAs or the Cyclone series using enhanced configuration devices, Altera recommends enabling decompression in Stratix II FPGAS or the Cyclone series only for faster configuration.

The compression algorithm used in Altera devices is optimized for FPGA configuration bitstreams. Since FPGAs have several layers of routing structures (for high performance and easy routability), large amounts of resources go unused. These unused routing and logic resources as well as un-initialized memory structures result in a large number of configuration RAM bits in the disabled state. Altera's proprietary compression algorithm takes advantage of such bitstream qualities.

The general guideline for effectiveness of compression is the higher the device logic/routing utilization, the lower the compression ratio (where the compression ratio is defined as the original bitstream size divided by the compressed bitstream size).

For Stratix designs, based on a suite of designs with varying amounts of logic utilization, the minimum compression ratio was observed to be 1.9 or a ~47% size reduction for these designs. Table 1–6 lists sample compression ratios from a suite of Stratix designs. These numbers serve as a guideline (not a specification) to help you allocate sufficient configuration memory to store compressed bitstreams.

Table 1–6. Stratix Compression Ratios ***(Note 1)***

Item	Minimum	Average
Logic Utilization	98%	64%
Compression Ratio	1.9	2.3
% Size Reduction	47%	57%

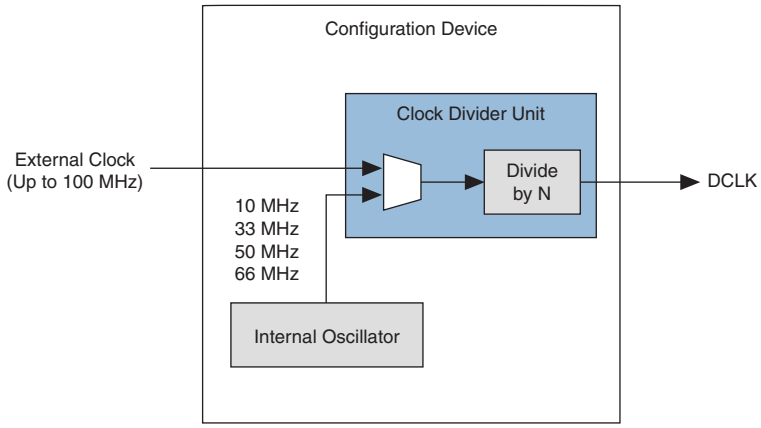
Note to Table 1–6:

(1) These numbers are preliminary. They are intended to serve as a guideline, not a specification.

Programmable Configuration Clock

The configuration clock (DCLK) speed is user programmable. One of two clock sources can be used to synthesize the configuration clock; a programmable oscillator or an external clock input pin (EXCLK). The configuration clock frequency can be further synthesized using the clock divider circuitry. This clock can be divided by the N counter to generate your DCLK output. The N divider supports all integer dividers between 1 and 16, as well as a 1.5 divider and a 2.5 divider. The duty cycle for all clock divisions other than non-integer divisions is 50% (for the non-integer dividers, the duty cycle will not be 50%). Refer to Figure 1–5 for a block diagram of the clock divider unit.

Figure 1–5. Clock Divider Unit



The DCLK frequency is limited by the maximum DCLK frequency the FPGA supports.

The maximum DCLK input frequency supported by the FGPA is specified in the appropriate FPGA family chapter in the *Configuration Handbook*.

The controller chip features a programmable oscillator that can output four different frequencies. The various settings generate clock outputs at frequencies as high as 10, 33, 50, and 66 MHz, as listed in Table 1–7.

Table 1–7. Internal Oscillator Frequencies

Frequency Setting	Min (MHz)	Typ (MHz)	Max (MHz)
10	6.4	8.0	10.0
33	21.0	26.5	33.0
50	32.0	40.0	50.0
66	42.0	53.0	66.0

Clock source, oscillator frequency, and clock divider (N) settings can be made in the Quartus II software, by accessing the **Configuration Device Options** inside the **Device Settings** window or the **Convert Programming Files** window. The same window can be used to select between the internal oscillator and the external clock (EXCLK) input pin as your configuration clock source. The default setting selects the internal oscillator at the 10 MHz setting as the clock source, with a divide factor of 1.

For more information about making the configuration clock source, frequency, and divider settings, refer to the *Altera Enhanced Configuration Devices* chapter in volume 2 of the *Configuration Handbook*.

Flash In-System Programming (ISP)

The flash memory inside enhanced configuration devices can be programmed in-system via the JTAG interface and the external flash interface. JTAG-based programming is facilitated by the configuration controller in the enhanced configuration device. External flash interface programming requires an external processor or FPGA to control the flash.

The enhanced configuration device flash memory supports 100,000 erase cycles.

JTAG-based Programming

The IEEE Std. 1149.1 JTAG Boundary Scan is implemented in enhanced configuration devices to facilitate the testing of its interconnection and functionality. Enhanced configuration devices also support the ISP mode. The enhanced configuration device is compliant with the IEEE Std. 1532 draft 2.0 specification.

The JTAG unit of the configuration controller communicates directly with the flash memory. The controller processes the ISP instructions and performs the necessary flash operations. The enhanced configuration devices support a maximum JTAG TCK frequency of 10 MHz.

During JTAG-based ISP, the external flash interface is not available. Before the JTAG interface programs the flash memory, an optional JTAG instruction (PENDCFG) can be used to assert the FPGA's *n*CONFIG pin (via the *n*INIT_CONF pin). This will keep the FPGA in reset and terminate any internal flash access. This function prevents contention on the flash pins when both JTAG ISP and an external FPGA or processor try to access the flash simultaneously. The *n*INIT_CONF pin is released when the initiate configuration (*n*INIT_CONF) JTAG instruction is updated. As a result, the FPGA is configured with the new configuration data stored in flash.

An initiate configuration (*n*INIT_CONF) JTAG instruction can be added to your programming file in the Quartus II software by enabling the **Initiate configuration after programming** option in the **Programmer options** window (Options menu).

Programming via External Flash Interface

This method allows parallel programming of the flash memory (using the 16-bit data bus). An external processor or FPGA acts as the flash controller and has access to programming data (via a communication link such as UART, Ethernet, and PCI). In addition to the program, erase, and verify operations, the external flash interface supports block/sector protection instructions.

For information about protection commands, areas, and lock bits, refer to the appropriate flash data sheets.

- For Micron flash-based EPC4, refer to the *Micron Flash Memory MT28F400B3 Data Sheet* at www.micron.com.
- For Sharp flash-based EPC16, refer to the *Sharp LHF16J06 Data Sheet Flash Memory Used in EPC16 Devices* at www.sharpsma.com.
- For the *Intel Advanced Boot Block Flash Memory (B3) 28F008/800B3, 28F016/160B3, 28F320B3, 28F640B3 Datasheet*, visit www.intel.com.

External flash interface programming is only allowed when the configuration controller has relinquished flash access (by tri-stating its internal interface). If the controller has not relinquished flash access (during configuration or JTAG-based ISP), you must hold the controller in reset before initiating external programming. The controller can be reset by holding the FPGA *n*CONFIG line at a logic low level. This keeps the controller in reset by holding the *n*STATUS-OE line low, allowing external flash access.

If initial programming of the enhanced configuration device is done in-system via the external flash interface, the controller must be kept in reset by driving the FPGA *n*CONFIG line low to prevent contention on the flash interface.

Pin Description

Table 1–8 through Table 1–10 describe the enhanced configuration device pins. These tables include configuration interface pins, external flash interface pins, JTAG interface pins, and other pins.

Table 1–8. Configuration Interface Pins

Pin Name	Pin Type	Description
DATA[7..0]	Output	Configuration data output bus. DATA changes on each falling edge of DCLK. DATA is latched into the FPGA on the rising edge of DCLK.
DCLK	Output	The DCLK output pin from the enhanced configuration device serves as the FPGA configuration clock. DATA is latched by the FPGA on the rising edge of DCLK.
*n*CS	Input	The *n*CS pin is an input to the enhanced configuration device and is connected to the FPGA's CONF_DONE signal for error detection after all configuration data is transmitted to the FPGA. The FPGA will always drive *n*CS and OE low when *n*CONFIG is asserted. This pin contains a programmable internal weak pull-up resistor of 6K Ω that can be disabled/enabled in the Quartus II software through the **Disable nCS and OE pull-ups on configuration device** option.
*n*INIT_CONF	Open-Drain Output	The *n*INIT_CONF pin can be connected to the *n*CONFIG pin on the FPGA to initiate configuration from the enhanced configuration device via a private JTAG instruction. This pin contains an internal weak pull-up resistor of 6K Ω that is always active. The INIT_CONF pin does not need to be connected if its functionality is not used. If *n*INIT_CONF is not used, *n*CONFIG must be pulled to V_{CC} either directly or through a pull-up resistor.
OE	Open-Drain Bidirectional	This pin is driven low when POR is not complete. A user-selectable 2-ms or 100-ms counter holds off the release of OE during initial power up to permit voltage levels to stabilize. POR time can be extended by externally holding OE low. OE is connected to the FPGA *n*STATUS signal. After the enhanced configuration device controller releases OE, it waits for the *n*STATUS-OE line to go high before starting the FPGA configuration process. This pin contains a programmable internal weak pull-up resistor of 6K Ω that can be disabled/enabled in the Quartus II software through the **Disable *n*CS** and OE pull-**ups on configuration device** option.

Table 1–9. External Flash Interface Pins (Part 1 of 2)

Pin Name	Pin Type	Description
A[20..0]	Input	These pins are the address input to the flash memory for read and write operations. The addresses are internally latched during a write cycle. When the external flash interface is not used, leave these pins floating (with a few exceptions *(1)*). These flash address, data, and control pins are internally connected to the configuration controller. In the 100-pin PQFP package, four address pins (A0, A1, A15, A16) are not internally connected to the controller. These loop-back connections must be made on the board between the C-A[] and F-A[] pins even when not using the external flash interface. All other address pins are connected internal to the package. All address pins are connected internally in the 88-pin UFBGA package. Pin A20 in EPC16 devices, pins A20 and A19 in EPC8 devices, and pins A20, A19, and A18 in EPC4 devices are no-connects. These pins should be left floating on the board.
DQ[15..0]	Bidirectional	This is the flash data bus interface between the flash memory and the controller. The controller or an external source drives DQ[15..0] during the flash command and the data write bus cycles. During the data read cycle, the flash memory drives the DQ[15..0] to the controller or external device. Leave these pins floating on the board when the external flash interface is not used.
CE#	Input	Active low flash input pin that activates the flash memory when asserted. When it is high, it deselects the device and reduces power consumption to standby levels. This flash input pin is internally connected to the controller. Leave this pin floating on the board when the external flash interface is not used.
RP# *(1)*	Input	Active low flash input pin that resets the flash when asserted. When high, it enables normal operation. When low, it inhibits write operation to the flash memory, which provides data protection during power transitions. This flash input is not internally connected to the controller. Hence, an external loop-back connection between C-RP# and F-RP# must be made on the board even when you are not using the external flash interface. When using the external flash interface, connect the external device to the RP# pin with the loop back. Always tri-state RP# when the flash is not in use.
OE#	Input	Active-low flash-control input that is asserted by the controller or external device during flash read cycles. When asserted, it enables the drivers of the flash output pins. Leave this pin floating on the board when the external flash interface is not used.
WE# *(1)*	Input	Active-low flash-write strobe asserted by the controller or external device during flash write cycles. When asserted, it controls writes to the flash memory. In the flash memory, addresses and data are latched on the rising edge of the WE# pulse. This flash input is not internally connected to the controller. Hence, an external loop-back connection between C-WE# and F-WE# must be made on the board even when you are not using the external flash interface. When using the external flash interface, connect the external device to the WE# pin with the loop back.

Table 1–9. External Flash Interface Pins (Part 2 of 2)

Pin Name	Pin Type	Description
WP#	Input	Usually tied to V_{CC} or ground on the board. The controller does not drive this pin because it could cause contention. Connection to V_{CC} is recommended for faster block erase/programming times and to allow programming of the flash-bottom boot block, which is required when programming the device using the Quartus II software. This pin should be connected to V_{CC} even when the external flash interface is not used.
V_{CCW}	Supply	Block erase, full-chip erase, word write, or lock-bit configuration power supply. Connect this pin to the 3.3-V V_{CC} supply, even when you are not using the external flash interface.
RY/BY#	Open-Drain Output	Flash asserts this pin when a write or erase operation is complete. This pin is not connected to the controller. RY/BY# is only available in Sharp flash-based EPC8 and EPC16. *(2)* Leave this pin floating when the external flash interface is not used.
BYTE#	Input	Flash byte-enable pin and is only available for enhanced configuration devices in the 100-pin PQFP package. This pin must be connected to V_{CC} on the board even when you are not using the external flash interface (the controller uses the flash in 16-bit mode). For Intel flash-based EPC device, this pin is connected to the V_{CCQ} of the Intel flash die internally. Therefore, BYTE# must be connected directly to V_{CC} without using any pull-up resistor.

Notes to Table 1–9:

(1) These pins can be driven to 12 V during production testing of the flash memory. Since the controller cannot tolerate the 12-V level, connections from the controller to these pins are not made internal to the package. Instead they are available as two separate pins. You must connect the two pins at the board level (for example, on the printed circuit board (PCB), connect the C-WE# pin from controller to F-WE# pin from the flash memory).

(2) For more information, refer to the Process Change Notification *PCN0506: Addition of Intel Flash Memory As Source For EPC4, EPC8 and EPC16 Enhanced Configuration Devices* and *Using the Intel Flash Memory-Based EPC4, EPC8 and EPC16* white paper.

Table 1–10. JTAG Interface Pins and Other Required Controller Pins (Part 1 of 2)

Pin Name	Pin Type	Description
TDI	Input	JTAG data input pin. Connect this pin to V_{CC} if the JTAG circuitry is not used.
TDO	Output	JTAG data output pin. Do not connect this pin if the JTAG circuitry is not used (leave floating).
TCK	Input	JTAG clock pin. Connect this pin to GND if the JTAG circuitry is not used.
TMS	Input	JTAG mode select pin. Connect this pin to V_{CC} if the JTAG circuitry is not used.
PGM[2..0]	Input	These three input pins select one of the eight pages of configuration data to configure the FPGAs in the system. Connect these pins on the board to select the page specified in the Quartus II software when generating the enhanced configuration device POF. PGM[2] is the MSB. The default selection is page 0; PGM[2..0]=000. These pins must not be left floating.

Table 1–10. JTAG Interface Pins and Other Required Controller Pins (Part 2 of 2)

Pin Name	Pin Type	Description
EXCLK	Input	Optional external clock input pin that can be used to generate the configuration clock (DCLK). When an external clock source is not used, connect this pin to a valid logic level (high or low) to prevent a floating-input buffer. If EXCLK is used, toggling the EXCLK input pin after the FPGA enters user mode will not effect the enhanced configuration device operation.
PORSEL	Input	This pin selects a 2-ms or 100-ms POR counter delay during power up. When PORSEL is low, POR time is 100 ms. When PORSEL is high, POR time is 2 ms. This pin must be connected to a valid logic level.
TM0	Input	For normal operation, this test pin must be connected to GND.
TM1	Input	For normal operating, this test pin must be connected to V_{CC}.

Power-On Reset

The POR circuit keeps the system in reset until power-supply voltage levels have stabilized. The POR time consists of the V_{CC} ramp time and a user-programmable POR delay counter. When the supply is stable and the POR counter expires, the POR circuit releases the OE pin. The POR time can be further extended by an external device by driving the OE pin low.

Do not execute JTAG or ISP instructions until POR is complete.

The enhanced configuration device supports a programmable POR delay setting. You can set the POR delay to the default 100-ms setting or reduce the POR delay to 2 ms for systems that require fast power-up. The PORSEL input pin controls this POR delay; a logic-high level selects the 2-ms delay, while a logic-low level selects the 100-ms delay.

The enhanced configuration device can enter reset under the following conditions:

- The POR reset starts at initial power-up during V_{CC} ramp-up or if V_{CC} drops below the minimum operating condition anytime after V_{CC} has stabilized
- The FPGA initiates reconfiguration by driving *n*STATUS low, which occurs if the FPGA detects a CRC error or if the FPGA's *n*CONFIG input pin is asserted
- The controller detects a configuration error and asserts OE to begin re-configuration of the Altera FPGA (for example, when CONF_DONE stays low after all configuration data has been transmitted)

Power Sequencing

Altera requires that you power-up the FPGA's V_{CCINT} supply before the enhanced configuration device's POR expires.

Power up needs to be controlled so that the enhanced configuration device's OE signal goes high after the CONF_DONE signal is pulled low. If the EEPC device exits POR before the FPGA is powered up, the CONF_DONE signal will be high because the pull-up resistor is holding this signal high. When the enhanced configuration device exits POR, OE is released and pulled high by a pull-up resistor. Since the enhanced configuration device samples the nCS signal on the rising edge of OE, it detects a high level on CONF_DONE and enters an idle mode. DATA and DCLK outputs will not toggle in this state and configuration will not begin. The enhanced configuration device will only exit this mode if it is powered down and then powered up correctly.

To ensure the enhanced configuration device enters configuration mode properly, you must ensure that the FPGA completes power-up before the enhanced configuration device exits POR.

The pin-selectable POR time feature is useful for ensuring this power-up sequence. The enhanced configuration device has two POR settings, 2 ms when PORSEL is set to a high level and 100 ms when PORSEL is set to a low level. For more margin, the 100-ms setting can be selected to allow the FPGA to power-up before configuration is attempted.

Alternatively, a power-monitoring circuit or a power-good signal can be used to keep the FPGA's nCONFIG pin asserted low until both supplies have stabilized. This ensures the correct power up sequence for successful configuration.

Programming and Configuration File Support

The Quartus II software provides programming support for the enhanced configuration device and automatically generates the **.pof** for the EPC4, EPC8, and EPC16 devices. In a multi-device project, the software can combine the **.sof** for multiple ACEX 1K, APEX 20K, APEX II, Cyclone series, FLEX 10K, Mercury, and Stratix series FPGAs into one programming file for the enhanced configuration device.

For details about generating programming files, refer to the *Altera Enhanced Configuration Devices* chapter and the *Software Settings* section in volume 2 of the *Configuration Handbook*.

Enhanced configuration devices can be programmed in-system through the industry-standard 4-pin JTAG interface. The ISP feature in the enhanced configuration device provides ease in prototyping and updating FPGA functionality.

After programming an enhanced configuration device in-system, FPGA configuration can be initiated by including the enhanced configuration device's JTAG INIT_CONF instruction (Table 1–11).

The ISP circuitry in the enhanced configuration device is compliant with the IEEE Std. 1532 specification. The IEEE Std. 1532 is a standard that allows concurrent ISP between devices from multiple vendors.

Table 1–11. Enhanced Configuration Device JTAG Instructions ***(Note 1)***

JTAG Instruction	OPCODE	Description
SAMPLE/ PRELOAD	00 0101 0101	Allows a snapshot of the state of the enhanced configuration device pins to be captured and examined during normal device operation and permits an initial data pattern output at the device pins.
EXTEST	00 0000 0000	Allows the external circuitry and board-level interconnections to be tested by forcing a test pattern at the output pins and capturing results at the input pins.
BYPASS	11 1111 1111	Places the 1-bit bypass register between the TDI and the TDO pins, which allow the BST data to pass synchronously through a selected device to adjacent devices during normal device operation.
IDCODE	00 0101 1001	Selects the device IDCODE register and places it between TDI and TDO, allowing the device IDCODE to be serially shifted out to TDO. The device IDCODE for all enhanced configuration devices is the same and shown below: 0100A0DDh
USERCODE	00 0111 1001	Selects the USERCODE register and places it between TDI and TDO, allowing the USERCODE to be serially shifted out the TDO. The 32-bit USERCODE is a programmable user-defined pattern.
INIT_CONF	00 0110 0001	This function initiates the FPGA re-configuration process by pulsing the *n*INIT_CONF pin low, which is connected to the FPGA *n*CONFIG pin. After this instruction is updated, the *n*INIT_CONF pin is pulsed low when the JTAG state machine enters Run-Test/Idle state. The *n*INIT_CONF pin is then released and *n*CONFIG is pulled high by the resistor after the JTAG state machine goes out of Run-Test/Idle state. The FPGA configuration starts after *n*CONFIG goes high. As a result, the FPGA is configured with the new configuration data stored in flash via ISP. This function can be added to your programming file (**.pof**, **.jam**, **.jbc**) in the Quartus II software by enabling the **Initiate configuration after programming** option in the **Programmer options** window (Options menu).
PENDCFG	00 0110 0101	This optional function can be used to hold the *n*INIT_CONF pin low during JTAG-based ISP of the enhanced configuration device. This feature is useful when the external flash interface is controlled by an external FPGA/processor. This function prevents contention on the flash pins when both the controller and external device try to access the flash simultaneously. Before the enhanced configuration device's controller can access the flash memory, the external FPGA/processor needs to tri-state its interface to flash.This can be ensured by resetting the FPGA using the *n*INIT_CONF, which drives the *n*CONFIG pin and keeps the external FPGA/processor in the "reset" state. The *n*INIT_CONF pin is released when the initiate configuration (INIT_CONF) JTAG instruction is issued.

Note to Table 1–11:

(1) Enhanced configuration device instruction register length is 10 and boundary scan length is 174.

For more information about the enhanced configuration device JTAG support, refer to the BSDL files provided at the Altera website.

Enhanced configuration devices can also be programmed by third-party flash programmers or on-board processors using the external flash interface. Programming files (**.pof**) can be converted to an Intel HEX format file (**.hexout**) using the Quartus II **Convert Programming Files** utility, for use with the programmers or processors.

You can also program the enhanced configuration devices using the Quartus II software, the Altera Programming Unit (APU), and the appropriate configuration device programming adapter. Table 1–12 lists which programming adapter to use with each enhanced configuration device.

Table 1–12. Programming Adapters

Device	Package	Adapter
EPC16	88-pin UFBGA	PLMUEPC-88
	100-pin PQFP	PLMQEPC-100
EPC8	100-pin PQFP	PLMQEPC-100
EPC4	100-pin PQFP	PLMQEPC-100

IEEE Std. 1149.1 (JTAG) Boundary-Scan

The enhanced configuration device provides JTAG BST circuitry that complies with the IEEE Std. 1149.1-1990 specification. JTAG boundary-scan testing can be performed before or after configuration, but not during configuration.

Figure 1–6 shows the timing requirements for the JTAG signals.

Figure 1–6. JTAG Timing Waveforms

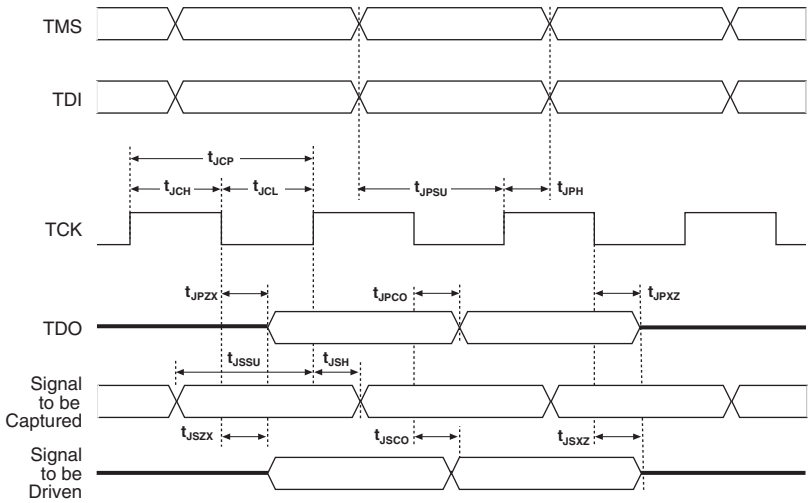


Table 1–13 lists the timing parameters and values for the enhanced configuration device.

Table 1–13. JTAG Timing Parameters and Values (Part 1 of 2)

Symbol	Parameter	Min	Max	Unit
t_{JCP}	TCK clock period	100	—	ns
t_{JCH}	TCK clock high time	50	—	ns
t_{JCL}	TCK clock low time	50	—	ns
t_{JPSU}	JTAG port setup time	20	—	ns
t_{JPH}	JTAG port hold time	45	—	ns

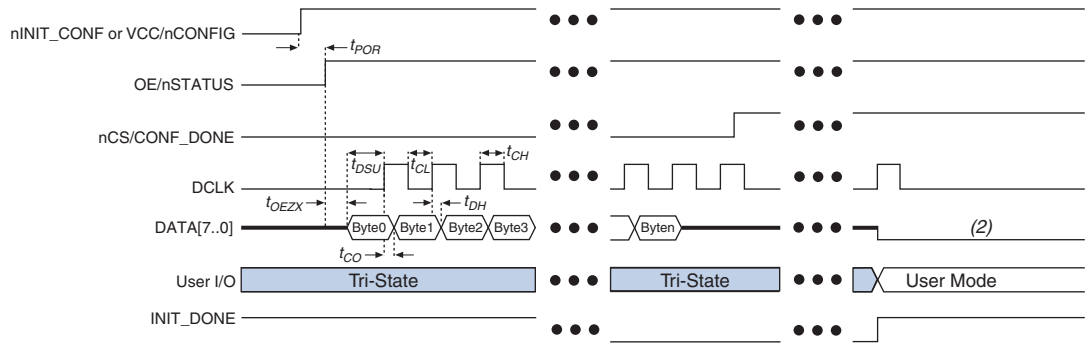
Table 1–13. JTAG Timing Parameters and Values (Part 2 of 2)

Symbol	Parameter	Min	Max	Unit
t_{JPCO}	JTAG port clock output	—	25	ns
t_{JPZX}	JTAG port high impedance to valid output	—	25	ns
t_{JPXZ}	JTAG port valid output to high impedance	—	25	ns
t_{JSSU}	Capture register setup time	20	—	ns
t_{JSH}	Capture register hold time	45	—	ns
t_{JSCO}	Update register clock to output	—	25	ns
t_{JSZX}	Update register high-impedance to valid output	—	25	ns
t_{JSXZ}	Update register valid output to high impedance	—	25	ns

Timing Information

Figure 1–7 shows the configuration timing waveform when using an enhanced configuration device.

Figure 1–7. Configuration Timing Waveform Using an Enhanced Configuration Device



Notes to Figure 1–7:

(1) The enhanced configuration device will drive `DCLK` low after configuration.

(2) The enhanced configuration device will drive `DATA[]` high after configuration.

Table 1–14 defines the timing parameters when using the enhanced configuration devices.

For more information about the flash memory (external flash interface) timing, refer to the appropriate flash data sheet on the Altera website at www.altera.com.

- For Micron flash-based EPC4, refer to the *Micron MT28F400B3 Data Sheet Flash Memory Used in EPC4 Devices* at www.micron.com.
- For Sharp flash-based EPC16, refer to the *Sharp LHF16J06 Data Sheet Flash Memory Used in EPC16 Devices* at www.sharpsma.com.

- For Intel flash-based EPC4 and EPC16, refer to Intel Flash 28F016B3 at www.intel.com.

Table 1–14. Enhanced Configuration Device Configuration Parameters

Symbol	Parameter	Condition	Min	Typ	Max	Unit
f_{DCLK}	DCLK frequency	40% duty cycle	—	—	66.7	MHz
t_{DCLK}	DCLK period	—	15	—	—	ns
t_{HC}	DCLK duty cycle high time	40% duty cycle	6	—	—	ns
t_{LC}	DCLK duty cycle low time	40% duty cycle	6	—	—	ns
t_{CE}	OE to first DCLK delay	—	40	—	—	ns
t_{OE}	OE to first DATA available	—	40	—	—	ns
t_{OH}	DCLK rising edge to DATA change	—	*(1)*	—	—	ns
t_{CF} *(2)*	OE assert to DCLK disable delay	—	277	—	—	ns
t_{DF} *(2)*	OE assert to DATA disable delay	—	277	—	—	ns
t_{RE} *(3)*	DCLK rising edge to OE	—	60	—	—	ns
t_{LOE}	OE assert time to assure reset	—	60	—	—	ns
f_{ECLK}	EXCLK input frequency	40% duty cycle	—	—	100	MHz
t_{ECLK}	EXCLK input period	—	10	—	—	ns
t_{ECLKH}	EXCLK input duty cycle high time	40% duty cycle	4	—	—	ns
t_{ECLKL}	EXCLK input duty cycle low time	40% duty cycle	4	—	—	ns
t_{ECLKR}	EXCLK input rise time	100 MHz	—	—	3	ns
t_{ECLKF}	EXCLK input fall time	100 MHz	—	—	3	ns
t_{POR} *(4)*	POR time	2 ms	1	2	3	ms
		100 ms	70	100	120	ms

Notes to Table 1–14:

(1) To calculate t_{OH}, use the following equation: t_{OH} = 0.5 (DCLK period) - 2.5 ns.
(2) This parameter is used for CRC error detection by the FPGA.
(3) This parameter is used for CONF_DONE error detection by the enhanced configuration device.
(4) The FPGA V_{CCINT} ramp time should be less than 1-ms for 2-ms POR, and it should be less than 70 ms for 100-ms POR.

Operating Conditions

Table 1–15 through Table 1–19 provide information about absolute maximum ratings, recommended operating conditions, DC operating conditions, supply current values, and pin capacitance data for the enhanced configuration devices.

Table 1–15. Enhanced Configuration Device Absolute Maximum Rating

Symbol	Parameter	Condition	Min	Max	Unit
V_{CC}	Supply voltage	With respect to ground	-0.2	4.6	V
V_I	DC input voltage	With respect to ground	-0.5	3.6	V
I_{MAX}	DC V_{CC} or ground current	—	—	100	mA
I_{OUT}	DC output current, per pin	—	-25	25	mA
P_D	Power dissipation	—	—	360	mW
T_{STG}	Storage temperature	No bias	-65	150	C
T_{AMB}	Ambient temperature	Under bias	-65	135	C
T_J	Junction temperature	Under bias	—	135	C

Table 1–16. Enhanced Configuration Device Recommended Operating Conditions

Symbol	Parameter	Condition	Min	Max	Unit
V_{CC}	Supplies voltage for 3.3-V operation	—	3.0	3.6	V
V_I	Input voltage	With respect to ground	–0.3	V_{CC} + 0.3	V
V_O	Output voltage	—	0	V_{CC}	V
T_A	Operating temperature	For commercial use	0	70	C
		For industrial use	–40	85	C
		For military use *(1)*	–55	125	C
T_R	Input rise time	—	—	20	ns
T_F	Input fall time	—	—	20	ns

Note to Table 1–16:

(1) Applicable for UBGA88 package of EPC16 device only.

Table 1–17. Enhanced Configuration Device DC Operating Conditions

Symbol	Parameter	Condition	Min	Typ	Max	Unit
V_{CC}	Supplies voltage to core	—	3.0	3.3	3.6	V
V_{IH}	High-level input voltage	—	2.0	—	V_{CC} + 0.3	V
V_{IL}	Low-level input voltage	—	—	—	0.8	V
V_{OH}	3.3-V mode high-level TTL output voltage	I_{OH} = –4 mA	2.4	—	—	V
	3.3-V mode high-level CMOS output voltage	I_{OH} = –0.1 mA	V_{CC} – 0.2	—	—	V
V_{OL}	Low-level output voltage TTL	I_{OL} = –4 mA DC	—	—	0.45	V
	Low-level output voltage CMOS	I_{OL} = –0.1 mA DC	—	—	0.2	V
I_I	Input leakage current	V_I = V_{CC} or ground	–10	—	10	µA
I_{OZ}	Tri-state output off-state current	V_O = V_{CC} or ground	–10	—	10	µA

Table 1–18. Enhanced Configuration Device I_{CC} Supply Current Values

Symbol	Parameter	Condition	Min	Typ	Max	Unit
I_{CC0}	Current (standby)	—	—	50	150	µA
I_{CC1}	V_{CC} supply current (during configuration)	—	—	60	90	mA
I_{CCW}	V_{CCW} supply current	—	—	*(1)*	*(1)*	—

Note to Table 1–18:

(1) For V_{CCW} supply current information, refer to the appropriate flash memory data sheet at **www.altera.com**.

Table 1–19. Enhanced Configuration Device Capacitance

Symbol	Parameter	Condition	Min	Max	Unit
CIN	Input pin capacitance	—	—	10	pF
COUT	Output pin capacitance	—	—	10	pF

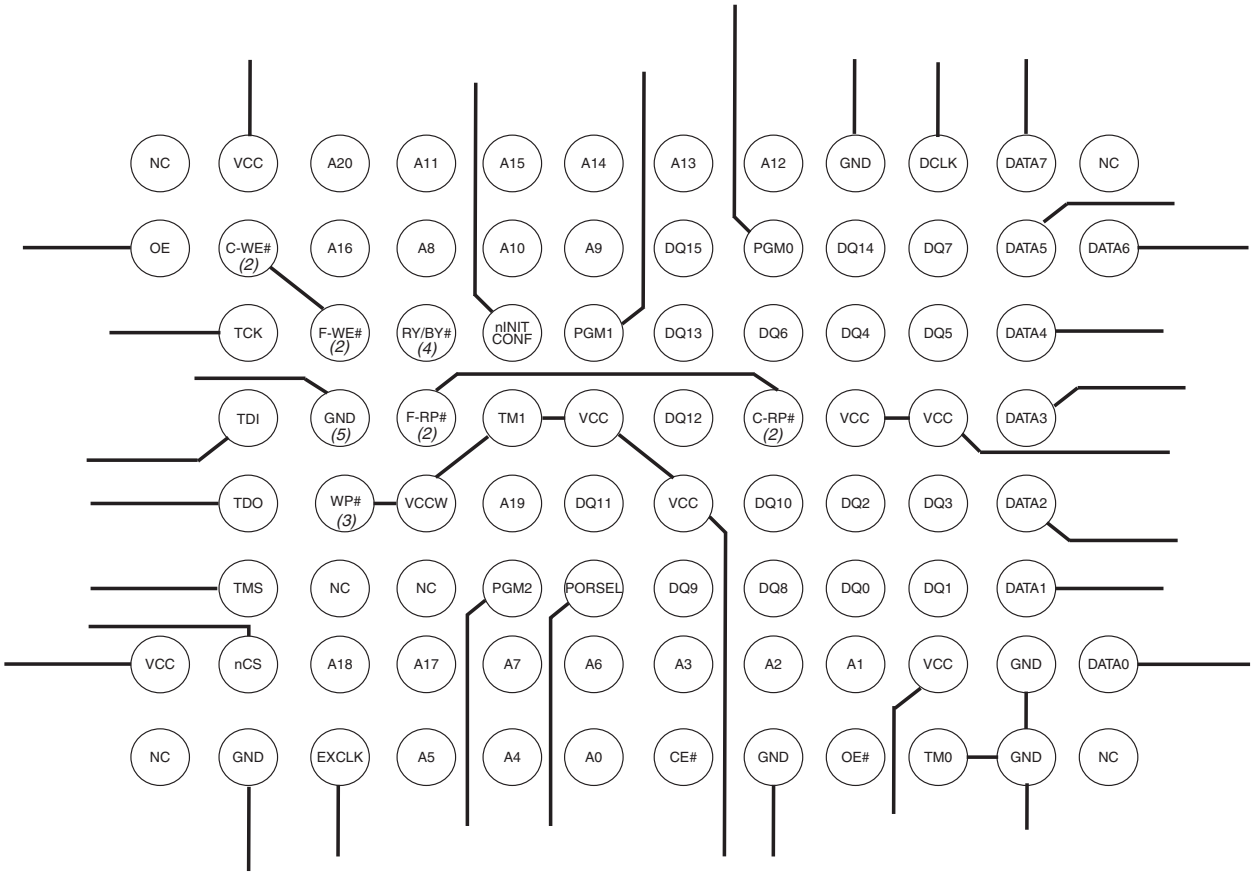
Package

The EPC16 enhanced configuration device is available in both the 88-pin UFBGA package and the 100-pin PQFP package. The UFBGA package, which is based on 0.8-mm ball pitch, maximizes board space efficiency. A board can be laid out for this package using a single PCB layer. The EPC8 and EPC4 devices are available in the 100-pin PQFP package.

Enhanced configuration devices support vertical migration in the 100-pin PQFP package.

Figure 1–8 shows the PCB routing for the 88-pin UFBGA package. The Gerber file for this layout is on the Altera website.

Figure 1–8. PCB Routing for 88-Pin UFBGA Package ***(Note 1)***



Notes to Figure 1–8:

(1) If the external flash interface feature is not used, then the flash pins should be left unconnected because they are internally connected to the controller unit. The only pins that need external connections are WP#, WE#, and RP#. If the flash is being used as an external memory source, then the flash pins should be connected as outlined in the pin descriptions section.

(2) F-RP# and F-WE# are pins on the flash die. C-RP# and C-WE# are pins on the controller die. C-WE# and F-WE# should be connected together on the PCB. F-RP# and C-RP# should also be connected together on the PCB.

(3) WP# (write protection pin) should be connected to a high level (3.3 V) to be able to program the flash bottom boot block, which is required when programming the device using the Quartus II software.

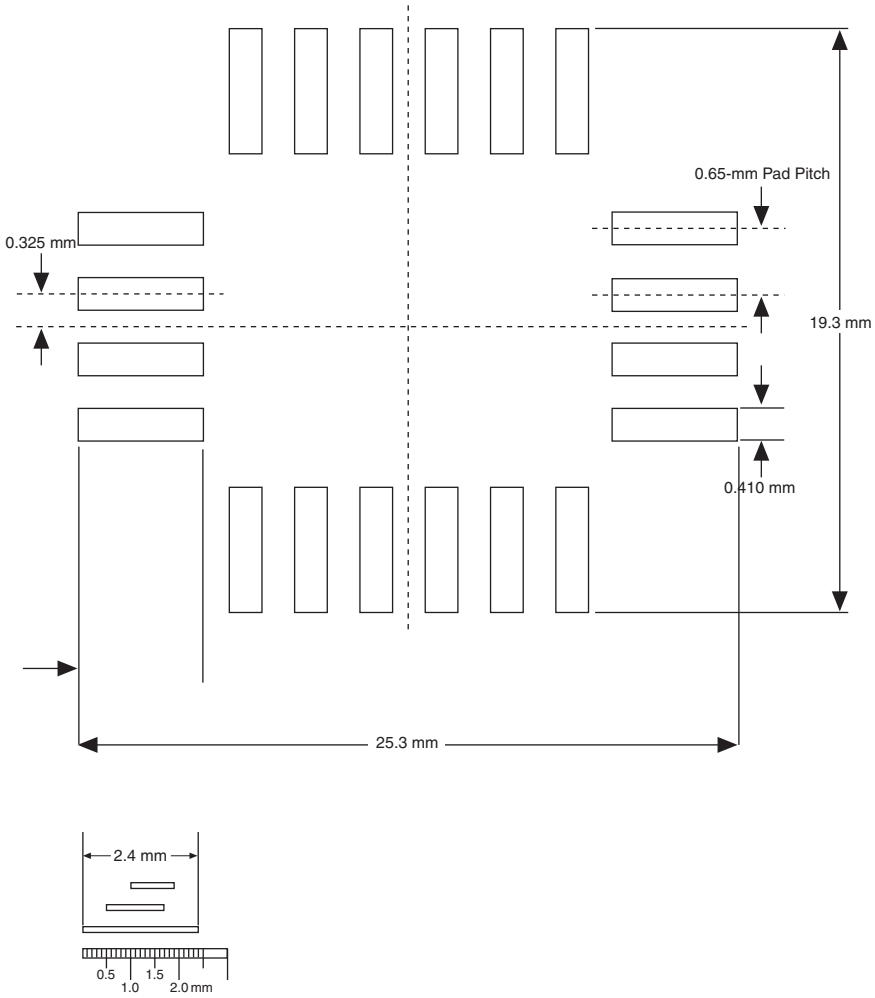
(4) RY/BY# is only available in Sharp flash-based enhanced configuration devices.

(5) Pin D3 is a NC pin for Intel Flash-based EPC16.

Package Layout Recommendation

Sharp flash-based EPC16 and EPC8 enhanced configuration devices in the 100-pin PQFP packages have different package dimensions than other Altera 100-pin PQFP devices (including the Micron flash-based EPC4, Intel flash-based EPC16, EPC8 and EPC4). Figure 1–9 shows the 100-pin PQFP PCB footprint specifications for enhanced configuration devices that allows for vertical migration between all devices. These footprint dimensions are based on vendor-supplied package outline diagrams.

Figure 1–9. Enhanced Configuration Device PCB Footprint Specifications for 100-Pin PQFP Packages *(Note 1)*, *(2)*



Notes to Figure 1–9:

(1) Used 0.5-mm increase for front and back of nominal foot length.

(2) Used 0.3-mm increase to maximum foot width.

For package outline drawings, refer to the *Altera Device Package Information Data Sheet*.

Device Pin-Outs

For pin-out information, refer to *Altera Configuration Devices Pin-Out Files*.

Document Revision History

Table 1–20 lists the revision history for this chapter.

Table 1–20. Chapter Revision History

Date	Version	Changes
June 2011	2.9	Updated Table 1–3 and Table 1–16.
December 2009	2.8	■ Added Table 1–1 and Table 1–2. ■ Updated Table 1–17 and Table 1–18. ■ Removed “Referenced Documents” section.
October 2008	2.7	■ Updated Table 2–1, Table 2–7, and Table 2–8. ■ Updated Figure 2–2, Figure 2–3, and Figure 2–4. ■ Updated “JTAG-based Programming” section. ■ Added “Intel-Flash-Based EPC Device Protection” section. ■ Updated new document format.
May 2008	2.6	Minor textual and style changes. Added “Referenced Documents” section.
February 2008	2.5	Updated Table 2–18 with information about EPC16UI88AA.
May 2007	2.4	Added “Intel-Flash-Based EPC Device Protection” section.
April 2007	2.3	Added document revision history.
October 2005	2.2	Made changes to content.
July 2004	2.0	■ Added Stratix II and Cyclone II device information throughout chapter. ■ Updated VCCW connection in Figure 2–2, Figure 2–3, and Figure 2–4. ■ Updated (Note 2) of Figure 2–2, Figure 2–3, and Figure 2–4. ■ Updated (Note 4) of Table 2–12. ■ Updated unit of ICC0 in Table 2–16. ■ Added ICCW to Table 2–16.
September 2003	1.0	Initial Release.

2. Altera Enhanced Configuration Devices

S52014-2.5

This chapter describes the latest enhanced configuration device flash memory standard with a feature-rich configuration controller. A single-chip configuration solution provides you with several new and advanced features that significantly reduce configuration times. This chapter discusses the hardware and software implementation of enhanced configuration device features such as concurrent and dynamic configuration, data compression, clock division, and an external flash memory interface. Enhanced configuration devices include EPC4, EPC8, and EPC16 devices.

Concurrent Configuration

Configuration data is transmitted from the enhanced configuration device to the SRAM-based device on the DATA lines. The DATA lines are outputs on the enhanced configuration devices, and inputs to the SRAM-based devices.

These DATA lines correspond to the Bit*n* lines in the **Convert Programming Files** window in the Altera® Quartus® II software. For example, if you specify a SRAM Object File (**.sof**) to use Bit0 in the Quartus II software, that **.sof** will be transmitted on the DATA[0] line from the enhanced configuration device to the SRAM-based device.

Supported Schemes and Guidelines

There are several different ways to configure Altera SRAM-based programmable logic devices (PLDs) with enhanced configuration devices:

- 1-bit passive serial (PS)
- 2-bit PS
- 4-bit PS
- 8-bit PS
- Fast passive parallel (FPP)

Additionally, you can use these configuration schemes in conjunction with the dynamic configuration option (previously called page mode operation) for sophisticated configuration setups.

FPP configuration mode uses the eight DATA[7..0] lines from the enhanced configuration device, which can be used to configure APEX™ II and Stratix® series devices. To decrease configuration time, FPP configuration provides eight bits of configuration data per clock cycle to the target device.

For more information about configuration schemes, refer to the *Enhanced Configuration Devices (EPC4, EPC8, and EPC16) Data Sheet*, or *Configuring Stratix & Stratix GX Devices*.

© 2009 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

For more information about additional enhanced configuration devices, refer to the Process Change Notification *PCN0506: Addition of Intel Flash Memory As Source For EPC4, EPC8 and EPC16 Enhanced Configuration Devices* and the *Using the Intel Flash Memory Based EPC4, EPC8, and EPC16 Devices* white paper.

Concurrent Configuration Using n-Bit PS Modes

The *n*-bit (*n* = 1, 2, 4, or 8) PS configuration mode allows enhanced configuration devices to concurrently configure SRAM-based devices or device chains. In addition, these devices do not have to be the same device family or density; they can be any combination of Altera SRAM-based devices. An individual enhanced configuration device DATA line is available for each targeted device. Each DATA line can also feed a daisy chain of devices.

The Quartus II software only allows the selection of *n*-bit PS configuration modes. However, you can use these modes to configure any number of devices from 1 to 8. When configuring SRAM-based devices using *n*-bit PS modes, use Table 2–1 to select the appropriate configuration mode for the fastest configuration times.

Mode selection has an impact on the amount of memory used, as described in "Calculating the Size of Configuration Space" on page 2–14.

Table 2–1. Recommended Configuration Using *n*-Bit PS Modes

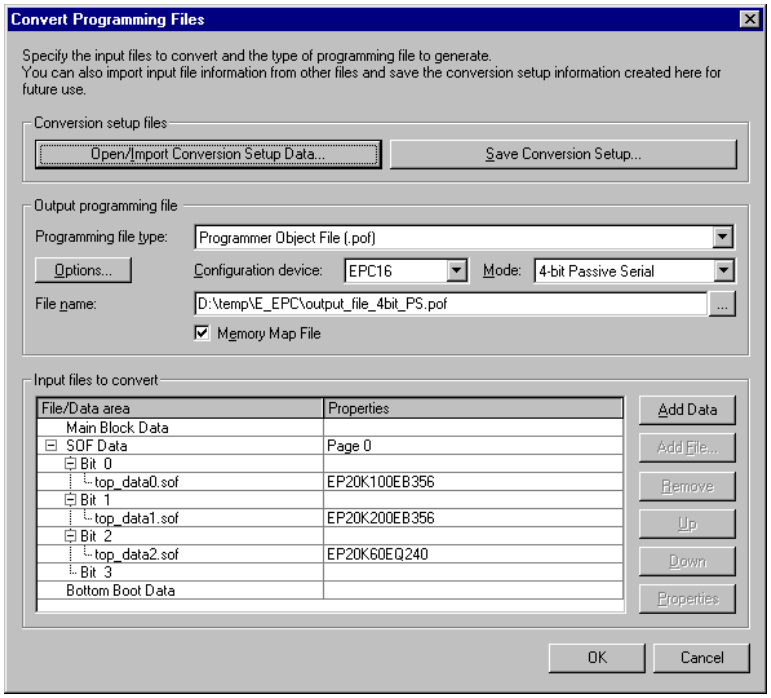
Number of Devices *(1)*	Recommended Configuration Mode
1	1-bit PS
2	2-bit PS
3	4-bit PS
4	4-bit PS
5	8-bit PS
6	8-bit PS
7	8-bit PS
8	8-bit PS

Note to Table 2–1:

(1) Assume that each DATA line is only configuring one device, not a daisy chain of devices.

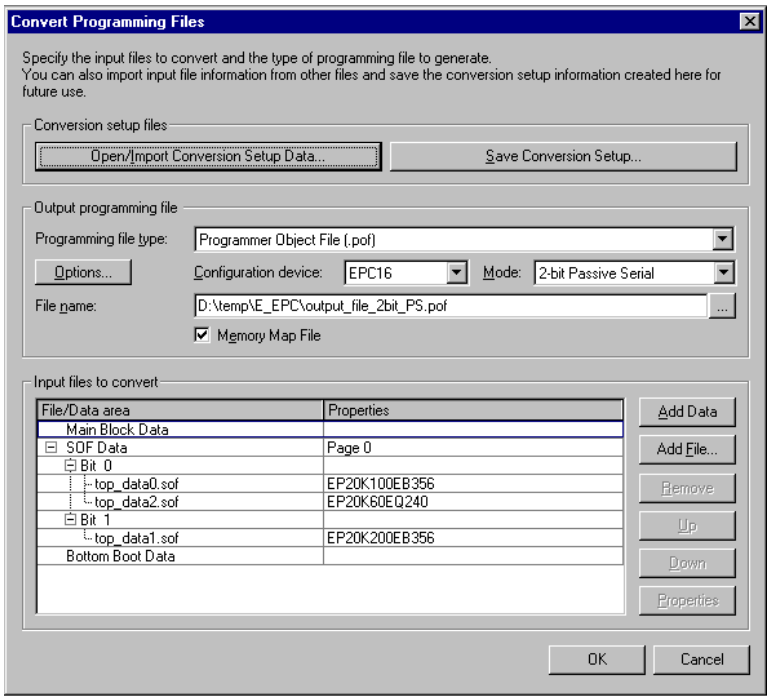
For example, if you configure three SRAM-based devices, you would use the 4-bit PS mode. For the DATA0, DATA1, and DATA2 lines, the corresponding **.sof** data will be transmitted from the configuration device to the SRAM-based PLD. For DATA3, you can leave the corresponding Bit3 line blank in the Quartus II software. On the printed circuit board (PCB), leave the DATA3 line from the enhanced configuration device unconnected. Figure 2–1 shows the **Quartus II Convert Programming Files** window (Tools menu) setup for this scheme.

Figure 2–1. Software Settings for Configuring Devices Using *n*-Bit PS Modes



Alternatively, you can daisy chain two SRAM-based devices to one DATA line while the other DATA lines drive one device each. For example, you could use the 2-bit PS mode to drive two SRAM-based devices with DATA Bit0 (EP20K100E and EP20K60E devices) and the third device (the EP20K200E device) with DATA Bit1. This 2-bit PS configuration scheme requires less space in the configuration flash memory, but may increase the total system configuration time as shown in Figure 2–2.

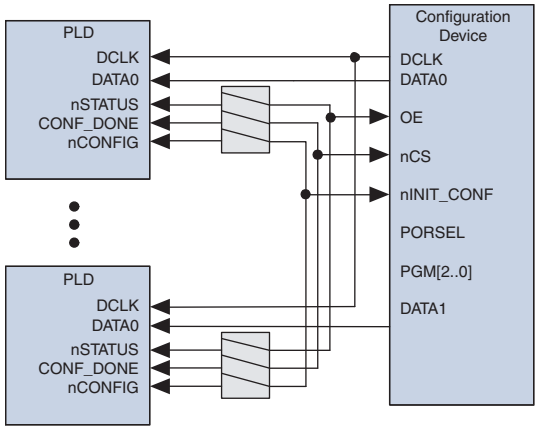
Figure 2–2. Daisy Chaining Two SRAM-Based Devices to One DATA Line



Design Guidelines

For debugging, Altera recommends keeping the control lines such as *n*STATUS, *n*CONFIG, and CONF_DONE between each PLD and the configuration device separate. You can keep control lines separate with a switch to manage which control signals are fed back into the enhanced configuration device. Figure 2–3 shows an example of the connections between the enhanced configuration device and the targeted PLDs.

Figure 2–3. Example of Using Debugging Switches for Control Lines



Dynamic Configuration (Page Mode) Implementation Overview

Pages in enhanced configuration devices allow you to organize and store various configurations for entire systems that use one or more Altera PLDs. This dynamic configuration (or page mode) feature allows systems to dynamically reconfigure their PLDs with different configuration files.

You can use different pages to store configuration files that support different standards (for example, I/O standards, memory). Alternatively, the different pages can place the system in different modes. For instance, page 0 could contain a configuration **.sof** for the PLD that only processes data packets; page 1 could contain a configuration file for the same PLD that processes data and voice packets.

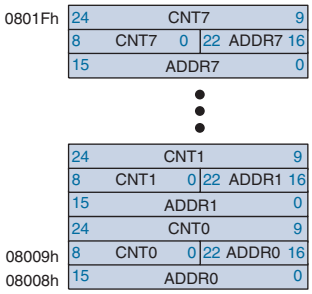
With the ability to dynamically switch pages, you can also configure Altera devices with various revisions for debugging without having to reprogram the configuration device. For example, you can configure a device that is on "stand-by" to perform another function and then reconfigure it back with the original configuration file.

A page is a section of the flash memory space that contains configuration data for all PLDs in the system. One page stores one system configuration regardless of the number of PLDs in the system. The size of each page is dynamic and can change each time the enhanced configuration device is reprogrammed. Enhanced configuration devices support a maximum of eight pages of configuration data, or eight system configurations. The number of pages is also limited to the density of the configuration device.

The number of pages required in a system is not dependent on the number of PLDs in the system, but depends on the number of unique system configurations.

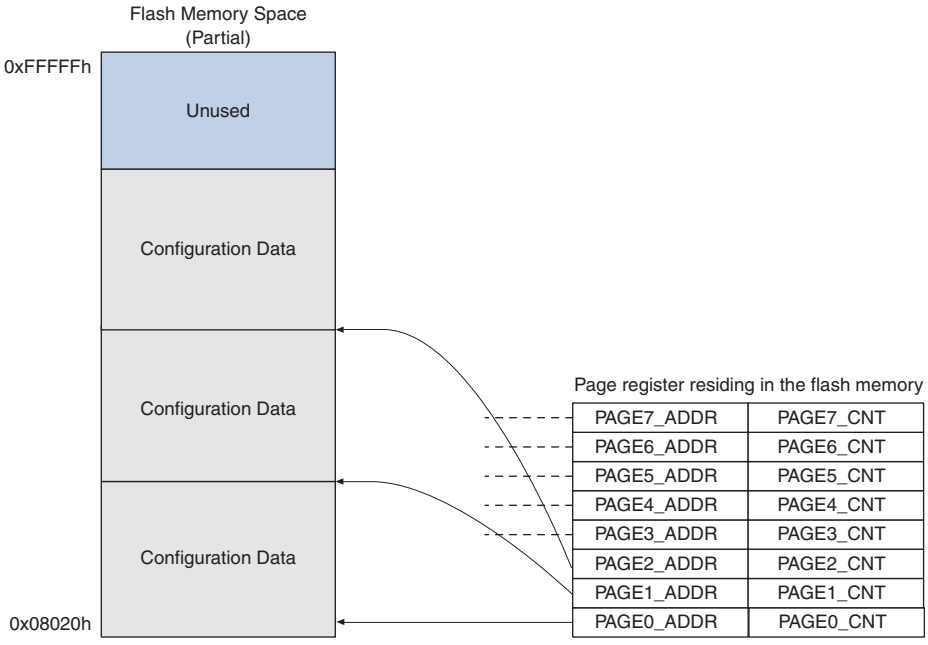
External page mode input pins PGM[2..0] determine which page to use during PLD configuration, and page pointers determine the data location. Each page pointer consists of a starting address register and a length count register. The word-addressable starting address register (23 bits) is used to determine where the page begins in the flash memory. The count register (25 bits) determines the length of the page counted in nibbles (group of 4 bits equaling half of a byte). Figure 2–4 shows a block diagram of the option-bit space and its address locations.

Figure 2–4. Option-Bit Memory Map



For example, a page for the EPC16 configuration device must start between word addresses 0x08020h and 0xFFFFFh and cannot overlap with other pages. Refer to Figure 2–5 for an EPC16 page mode example using three pages.

Figure 2–5. EPC16 Page Mode Implementation Example



During configuration, different pages are selected by the PGM[2..0] pins. These pins are used to select one out of eight pages (or eight system configurations). PGM[2..0] pins are sampled once before the configuration data is sent to the target PLDs.

Setting the PGM[2..0] pins to select an incorrect page (for example, a page that is non-existent or a blank page) causes the enhanced configuration device to enter an erroneous state. The only way to recover from this state is to set the PGM[2..0] pins to select a valid page and then power cycle the board.

To ensure proper configuration, only set the PGM[2..0]pins to select valid pages.

Within each page, you can store as many configuration files as your system needs. There is no limitation to the length of a page except for the physical limitation determined by the size of the flash memory (for example, 0xFFFFFFh for EPC16 devices). However, all pages must be contiguous.

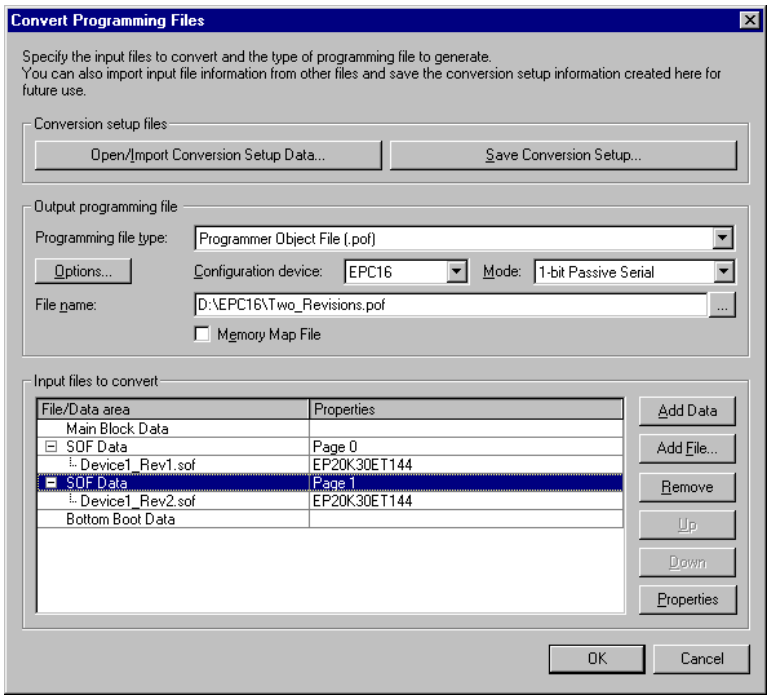
Software Implementation (Convert Programming Files)

The **Convert Programming Files** window (Tools menu) in the Quartus II software allows you to create enhanced configuration device programmer object files (**.pof**) and enable the dynamic configuration feature.

Passive parallel asynchronous (PPA) and passive parallel synchronous (PPS) configuration modes are not supported by enhanced configuration devices. If you choose one of these modes, the Quartus II software reports an error message when the **.pof** is generated.

In the **Convert Programming Files** window, there are **SOF Data** entries (**.sof**), located in the **Input files to convert** dialog box. Each **SOF Data** entry refers to a unique system configuration. Figure 2–6 shows the setup for a system that has one APEX device and uses two pages, 0 and 1. Each page has a different version of the configuration file for the same APEX device.

Figure 2–6. Using Page Mode Example



To set the page pointer to point to a particular page or **SOF Data** entry, select **SOF Data** and click **Properties**. Clicking **Properties** launches the **SOF Data Properties** window where you can select page pointers to point to the chosen **SOF Data**. If you do not use the **SOF Data Properties** window to make changes, the default page is 0. Each **SOF Data** entry for your configuration device must have a unique page number. Figure 2–7 shows page pointer 1 assigned to the **SOF Data** section containing **Device1_Rev2.sof** (Figure 2–6).

Figure 2–7. Software Setting for Selecting Pages

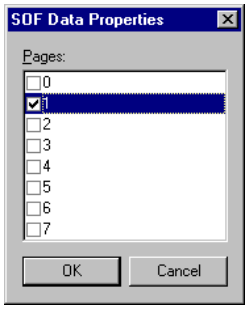
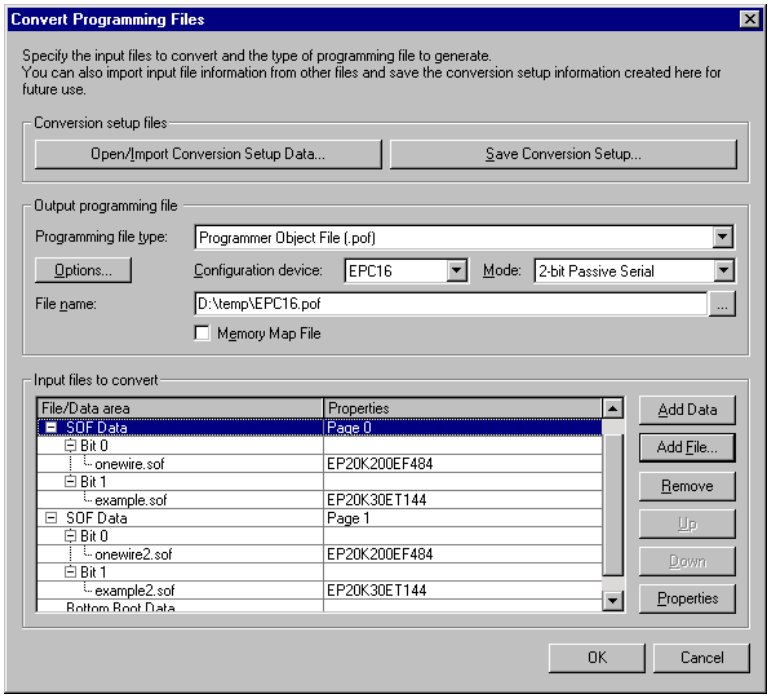


Figure 2–8 shows a more complex setup that uses the 2-bit PS configuration mode to concurrently configure two different APEX devices with multiple pages storing two revisions of each design. Two configurations for the entire system requires four configuration files (the number of devices multiplied by the number of unique system configurations).

Figure 2–8. Concurrent Configuration of Two Devices with Two System Configurations



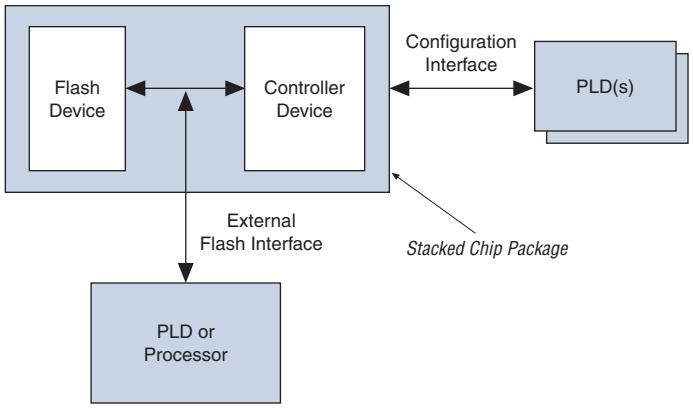
By selecting the **Memory Map File** option, the Quartus II Memory Map output file (**.map**) is generated, describing the flash memory address locations. This information is typically useful when using the external flash interface feature.

External Flash Memory Interface

Enhanced configuration devices support an external flash interface that allows devices external to the controller access to the enhanced configuration device's flash memory. You can use the flash memory to store boot or application code for processors, or as general-purpose memory for processors and PLDs.

Figure 2–9 shows the interfaces available on the enhanced configuration device.

Figure 2–9. Enhanced Configuration Device Interfaces



Applications that require remote update capabilities for on-board programmable logic (Stratix series devices), and applications that use soft-embedded processor cores (Nios® embedded processor) typically use the external flash memory interface feature.

For soft-embedded processor core applications, the controller configures the programmable logic with the configuration data stored in the flash memory. On successful configuration, the embedded processor uses the external flash interface to boot up and run code from the same flash memory, eliminating the need for a stand-alone flash memory device.

For applications requiring remote system configuration capabilities, a processor or PLD can use the external flash interface to store an updated configuration image into a new page in flash memory (the external flash interface coupled with dynamic configuration). You can obtain new configuration data from a local intelligent host or through the Internet. Reconfiguring the system with the new page updates the system configuration.

For more information about implementing remote and local system updates with enhanced configuration devices, refer to the *Using Remote System Configuration with Stratix & Stratix GX Devices* chapter in the *Stratix Handbook*, or the *Remote System Upgrades with Stratix II & Stratix II GX Devices* chapter in the *Stratix II Handbook*.

Currently, EPC4 and EPC16 configuration devices support the external flash interface. For support of this feature in other enhanced configuration devices, contact Altera Applications.

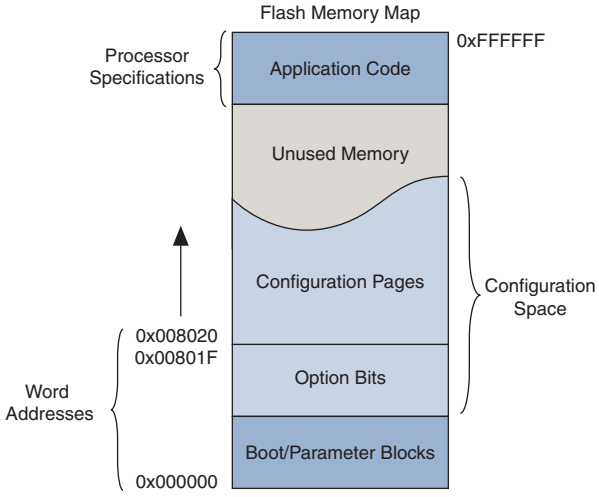
Flash Memory Map

You can divide an enhanced configuration device's flash memory into two categories: logical (configuration and processor space) and physical (flash data block boundaries). Configuration space consists of portions of memory used to store configuration option bits and configuration data. Processor space consists of portions of memory used to store boot and application code.

Logical Divisions

In all enhanced configuration devices, configuration option bits are stored ranging from word address `0x008000` to `0x00801F` (byte address `0x010000` to `0x01003F`). These bits are used to enable various controller features such as configuration mode selection, compression mode selection, and clock divider selection. In all enhanced configuration devices, configuration data is stored starting from word address location `0x008020` or byte address `0x010040`. The ending address of configuration space is not fixed and depends on the number and density of PLDs configured using the enhanced configuration device as well as the number of pages. All remaining address locations above the configuration space are available for processor application code. The boot space spans addresses `0x000000` to `0x007FFF`. Both boot and application code spaces are intended for use by an external processor or PLD. Figure 2–10 shows the flash memory map inside an EPC16 device.

Figure 2–10. EPC16 Flash Memory Map



Physical Divisions

Conversely, physical divisions are flash data blocks that can be individually written to and erased. For instance, the Sharp flash-based EPC16 device contains 16-Mbit Sharp flash memory that is divided into 2 boot blocks, 6 parameter blocks, and 31 main data blocks. These physical divisions vary from one flash memory or vendor to another and must be considered if the external flash interface is used to erase or write flash memory. These divisions are not significant if the interface is used as a read-only interface after initial programming.

For detailed information about enhanced configuration device flash memories, refer to the following documents:

- For Micron flash-based EPC4, refer to the *Micron Flash Memory MT28F400B3 Data Sheet* at **www.micron.com**.
- For Sharp flash-based EPC16, refer to the *Sharp LHF16J06 Data Sheet Flash Memory Used in EPC16 Devices* at **www.sharpsma.com**.
- For the *Intel Advanced Boot Block Flash Memory (B3) 28F008/800B3, 28F016/160B3, 28F320B3, 28F640B3 Datasheet*, visit **www.intel.com**.

Interface Availability and Connections

Flash memory ports are shared between the internal controller and the external device. A processor or PLD can use the external flash interface to access flash memory only when the controller is not using the interface. Therefore, the internal controller is the primary master of the bus, while the external device is the secondary master.

Flash memory ports (address, data, and control) are internally connected to the controller device. Additionally, these ports are connected to pins on the package providing the external interface. During in-system programming of the enhanced configuration device as well as configuration of the PLDs, the controller uses the internal interface to flash memory, rendering the external interface unavailable. External devices should tri-state all connections (address, data, and control) for the entire duration of in-system programming and configuration to prevent contention.

On completion of in-system programming and configuration, the internal controller tri-states its interface to the flash memory and enables weak internal pull-up resistors on address and control lines as well as bus-hold circuits on the data lines. The internal flash interface is now disabled and the external flash interface is available.

If you do not use the external flash interface feature, most flash-related pins must be left unconnected on the board to avoid contention. There are a few exceptions to this guideline outlined in the data sheet and pin-out tables.

For detailed schematics, refer to the *Enhanced Configuration Devices (EPC4, EPC8, and EPC16) Data Sheet*.

Quartus II Software Support

You can use the **Convert Programming Files** window to generate flash memory programming files. You can program flash memory in-system using JTAG or through the external flash interface. Select the **.pof** when programming the flash memory in-system. You can also convert this **.pof** to a Jam™ standard test and programming language (STAPL) file (**.jam**) or Jam Byte-Code file (**.jbc**) for in-system programming. When programming the flash memory through the external flash interface, you can create a **.hexout** from this window.

☞ The **.hexout** used for programming enhanced configuration devices is different from the **.hexout** configuration file generated for SRAM PLDs.

Along with PLD configuration files, you can program processor boot and application code into flash memory through the **Convert Programming Files** window. You can add a **.hex** file containing boot code to the **Bottom Boot Data** section of the window. Similarly, you can add a **.hex** file containing application code to the **Main Block Data** section. You can store these files in the flash memory using relative or absolute addressing. For selecting the type of addressing, highlight the **Bottom Boot Data** or **Main Block Data** section and click **Properties (Convert Programming Files** window).

Relative addressing mode allows the Quartus II software to pick the location of the file in memory. For example, the Quartus II software always stores boot code starting at address location `0x000000`. This data increases to higher addresses.

☞ The maximum boot file size for the EPC16 configuration device is 32K words or 64 Kbytes. The boot code is limited to the boot and flash memory parameter blocks.

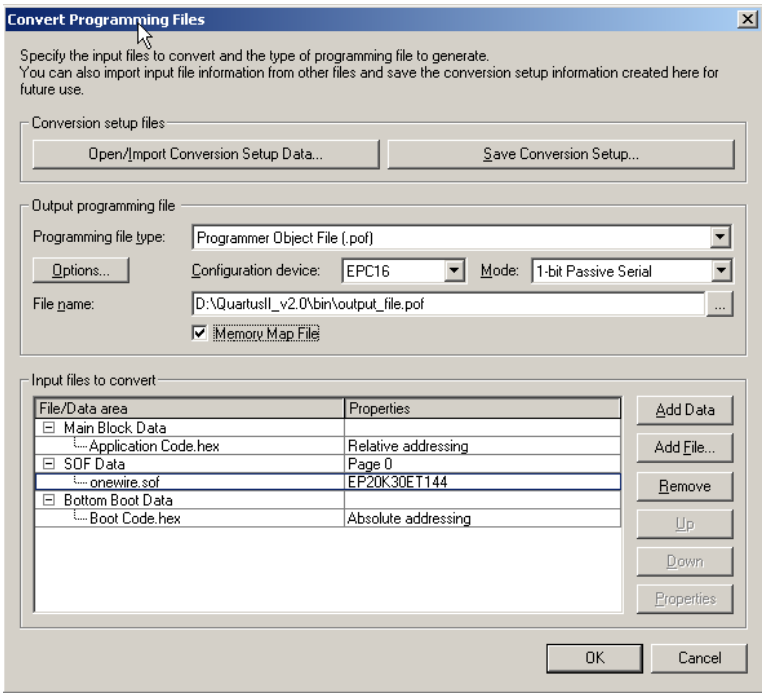
When you select relative addressing mode for **Main Block Data**, the Quartus II software aligns the last byte of information with the highest address (for example, `0x1FFFFF`). Therefore, the starting address is dependent on the size of the **.hex** file. You can easily obtain the starting address of the application code with the **.map** file discussed in this section.

Conversely, the absolute addressing mode forces the Quartus II software to store the boot or application **.hex** file data in address locations specified inside the **.hex** file itself. When this mode is selected, create **.hex** files with the correct offsets and ensure there is no overlap with addresses used for storing configuration data.

Figure 2–11 shows a screen shot of the **Convert Programming Files** window setup to create a **.pof** and **.map** file for an enhanced configuration device.

Only one **.hex** file can be added to the **Bottom Boot Data** and **Main Block Data** sections of this window.

Figure 2–11. Storing Boot and Application Code in Flash Memory



You can use the **Quartus II Convert Programming Files** window to create two files specific to the external flash interface feature—the **.hexout** and the **.map** files. The **.hexout** contains an image of the flash memory and the **.map** file contains memory map information. The **.hexout** can be used by an external processor or PLD to program the flash memory via the external flash interface. The **.map** file contains starting and ending addresses for boot code, configuration page data, and application code.

You can use the **.hexout** to program blank enhanced configuration devices, update portions of the flash memory (for example, a new configuration page), or both. This file uses the Intel hexadecimal file format and contains 16 Mbits or 2 Mbytes of data. Table 2–2 lists the format of the **.map** file.

Table 2–2. File Format (.map) ***(Note 1)***

Block	Start Address	End Address
BOTTOM BOOT	0x00000000	0x0000001F
OPTION BITS	0x00010000	0x0001003F
PAGE 0	0x00010040	0x0001AD7F
MAIN	0x001FFFE0	0x001FFFFF

Note to Table 2–2:

(1) All the addresses in this file are byte addresses.

To perform partial flash memory updates, select the relevant portions of the **.hexout** using memory map information provided in the **.map** file.

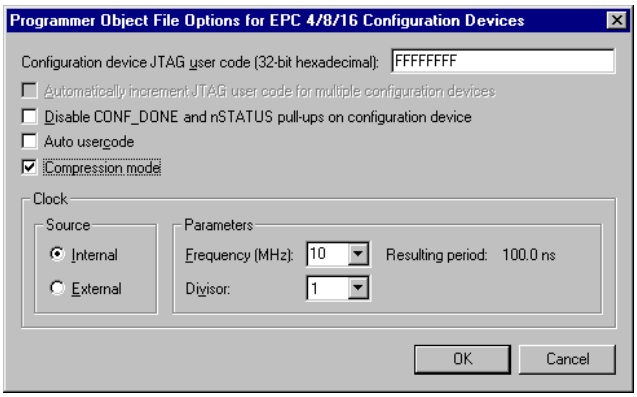
Configuration data and processor space data could exist within the same physical data block. In such cases, erasing the physical data block would affect both configuration and processor data, requiring you to update both. You can avoid this situation by storing application data starting from the next available whole data block.

Data Compression

Enhanced configuration devices support an efficient compression algorithm that compresses configuration data by 1.9× for typical designs, effectively doubling the size of the device. To select the right density for enhanced configuration devices, you should pre-calculate the total size of uncompressed configuration space.

By clicking **Options** (Convert Programming Files window), you can turn on the **Compression mode** option in the Programming Object File Options window with **pof** selected as the programming file type, as shown in Figure 2–12.

Figure 2–12. Selecting Compression Mode



Calculating the Size of Configuration Space

When using 1-bit PS configuration mode to serially configure multiple devices, all configuration data is transmitted through the same DATA line and the devices are daisy-chained together. Therefore, the total size of the uncompressed configuration data is equal to the sum of the SRAM-based device's configuration file size multiplied by the number of pages used.

When using *n*-bit PS configuration mode to concurrently configure multiple devices, each SRAM-based device has its own DATA line from the enhanced configuration devices. The total size of the uncompressed configuration space is equal to the size of the largest device's configuration file size multiplied by *n* (where *n* = 1, 2, 4, or 8), which is then multiplied by the number of pages used. For example, if three devices are concurrently configured using 4-bit PS configuration mode, the total size of the uncompressed configuration space is equal to the size of the largest device's configuration file multiplied by four.

When using FPP configuration mode, the total size of the uncompressed configuration space is equal to the sum of the SRAM-based device's configuration file size multiplied by the number of pages used.

For configuration file sizes of SRAM-based devices, refer to the *Configuration Handbook*.

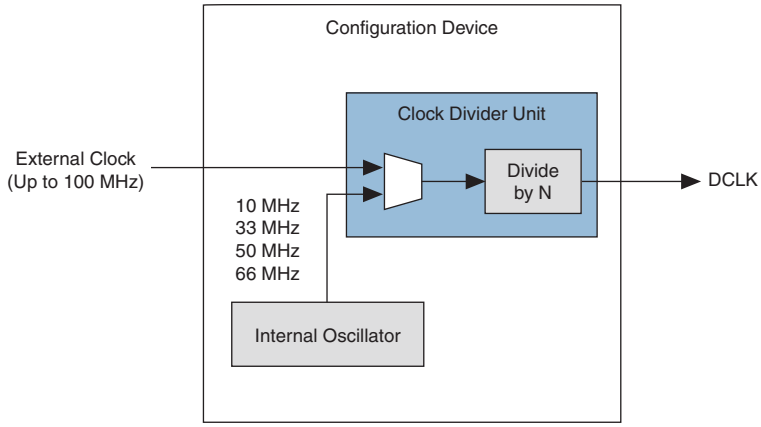
Clock Divider

The clock divider value specifies the clock frequency divisor, which is used to determine the DCLK frequency, or how fast the data is clocked into the SRAM-based device. You must consider the maximum DCLK input frequency of the targeted SRAM device family while selecting the clock input and divider settings. For DCLK timing specifications of SRAM-based devices, refer to the *Configuration Handbook*.

Settings and Guidelines

Enhanced configuration devices can use either the internal oscillator or an external clock source to clock data into SRAM-based devices, as shown in Figure 2–13. The enhanced configuration device's internal oscillator runs at nominal speeds of 10, 33, 50, or 66 MHz. The minimum and maximum speeds are shown in the *Enhanced Configuration Device Data Sheet*. Additionally, the enhanced configuration device can accept an external clock source running at speeds of up to 100 MHz.

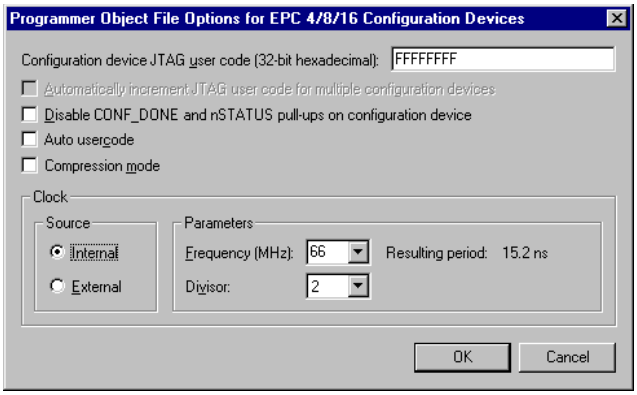
Figure 2–13. Clock Divider Unit in Enhanced Configuration Devices



Software Implementations

You can select the clock source and the clock speed in the **Programming Object File Options** window with **pof** selected as the programming file type (Convert Programming Files window), as shown in Figure 2–14. You can type the appropriate external clock frequency in the **Frequency (MHz)** drop-down menu, and select any value from the divisor list regardless of the clock source setting.

Figure 2–14. Software for Setting Clock Source and Clock Divisor



Conclusion

The enhanced configuration device is a single-chip configuration solution that provides increased configuration flexibility and faster time-to-market. Features such as data compression, multiple clock sources, clock division, and parallel or concurrent programming significantly reduce configuration times, while the dynamic configuration mode and the external flash interface take intelligent system configuration to a higher level.

Document Revision History

Table 2–3 lists the revision history for this chapter.

Table 2–3. Chapter Revision History

Date	Version	Changes
December 2009	2.5	Removed “Referenced Documents” section.
October 2008	2.4	■ Added “Referenced Documents” section. ■ Updated new document format.
April 2007	2.3	Added document revision history.
October 2005	2.2	Technical content added.
July 2004	2.0	■ Added text regarding pointing to an incorrect page after Figure 2–8. ■ Renamed **.hexpof** to **.hexout** throughout chapter.
September 2003	1.0	Initial Release.

3. Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet

C51014-3.4

Features

The serial configuration devices provide the following features:

- 1-, 4-, 16-, 64-, and 128-Mbit flash memory devices that serially configure Arria® series, Cyclone® series, all device families in the Stratix® series except the Stratix device family, and FPGAs using the active serial (AS) configuration scheme
- Easy-to-use four-pin interface
- Low cost, low-pin count, and non-volatile memory
- Low current during configuration and near-zero standby mode current
- 2.7-V to 3.6-V operation
- EPCS1 and EPCS4 available in 8-pin small outline integrated circuit (SOIC) package. EPCS16 available in 8-pin or 16-pin SOIC packages. EPCS64 and EPCS128 available in 16-pin SOIC package
- Enables the Nios® processor to access unused flash memory through AS memory interface
- Re-programmable memory with more than 100,000 erase/program cycles
- Write protection support for memory sectors using status register bits
- In-system programming support with SRunner software driver
- In-system programming support with USB Blaster™, EthernetBlaster, or ByteBlaster™ II download cables
- Additional programming support with the Altera® Programming Unit (APU) and programming hardware from BP Microsystems, System General, and other vendors
- Delivered with the memory array erased (all the bits set to `1`)

The term “serial configuration devices” used in this document refers to Altera EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128.

© 2011 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera’s standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

Functional Description

With SRAM-based devices that support active serial configuration, configuration data must be reloaded each time the device powers up, the system reconfigures, or when new configuration data is required. Serial configuration devices are flash memory devices with a serial interface that can store configuration data for FPGA devices that support active serial configuration and reload the data to the device upon power-up or reconfiguration. Table 3–1 summarizes the features of the Altera configuration devices and the amount of configuration space they hold.

Table 3–1. Altera Configuration Devices *(Note 1)*, *(2)*

Device	Memory Size (bits)	On-Chip Decompression Support	ISP Support	Cascading Support	Reprogrammable	Operating Voltage (V)
EPCS1	1,048,576	No	Yes	No	Yes	3.3
EPCS4	4,194,304	No	Yes	No	Yes	3.3
EPCS16	16,777,216	No	Yes	No	Yes	3.3
EPCS64	67,108,864	No	Yes	No	Yes	3.3
EPCS128	134,217,728	No	Yes	No	Yes	3.3

Notes to Table 3–1:

(1) To program these devices using Altera Programming Unit or Master Programming Unit, refer to *Altera Programming Hardware Data Sheet*.

(2) The EPCS device can be re-programmed in system with Byte Blaster II download cable or an external microprocessor using SRunner. For more information about SRunner, refer to the *AN418, SRunner: An Embedded Solution for EPCS Programming*.

For an 8-pin SOIC package, you can migrate vertically from the EPCS1 to the EPCS4 or EPCS16 because the EPCS devices are offered in the same device package. Similarly, for a 16-pin SOIC package, you can migrate vertically from the EPCS16 to the EPCS64 or EPCS128.

Use the compression ratio calculation to determine the FPGA device to fit the EPCS.

Example 3–1. Compression Ratio Calculation

EP4SGX530 = 189,000,000 bits

EPCS128 = 134,217,728 bits

Preliminary data indicates that compression typically reduces the configuration bitstream size by 35% to 55%. We take the worst case that is 35% compression.

```
189,000,000 bits x 0.65 = 122,850,000 bits
It fits EPCS128 device.
```

With the new data-decompression feature in Arria series, Cyclone series, and all device families in the Stratix series except the Stratix device family, you can use smaller serial configuration devices to configure larger FPGAs.

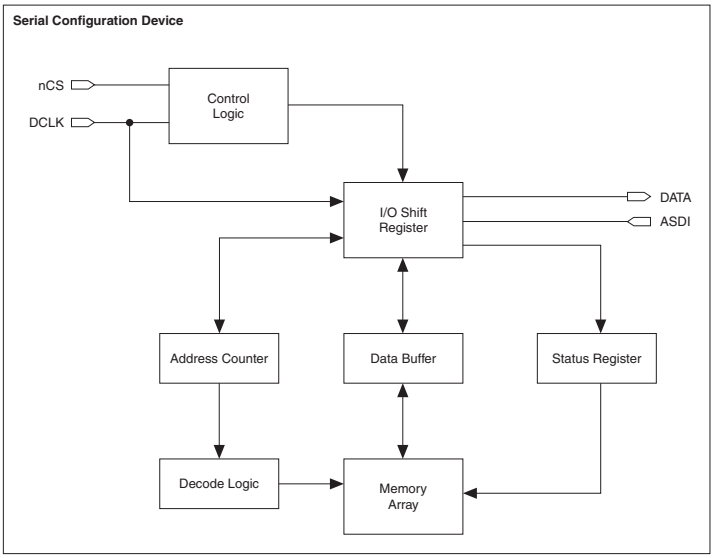
Serial configuration devices cannot be cascaded.

For more information about the FPGA decompression feature, refer to the configuration chapter in the appropriate device handbook.

The serial configuration devices are designed to configure the Cyclone series and all device families in the Stratix series except the Stratix device family. It cannot configure other existing Altera FPGA device families.

Figure 3–1 shows the serial configuration device block diagram.

Figure 3–1. Serial Configuration Device Block Diagram



Accessing Memory in Serial Configuration Devices

You can access the unused memory locations of the serial configuration device to store or retrieve data through the Nios processor and SOPC Builder. SOPC Builder is an Altera tool for creating bus-based (especially microprocessor-based) systems in Altera devices. SOPC Builder assembles library components such as processors and memories into custom microprocessor systems.

SOPC Builder includes the EPCS device controller core, which is an interface core specifically designed to work with the serial configuration device. With this core, you can create a system with a Nios embedded processor that allows software access to any memory location within the serial configuration device.

For more information about accessing memory within the serial configuration device, refer to the *Active Serial Memory Interface Data Sheet*.

Active Serial FPGA Configuration

The following Altera FPGAs support Active Serial (AS) configuration scheme with serial configuration devices:

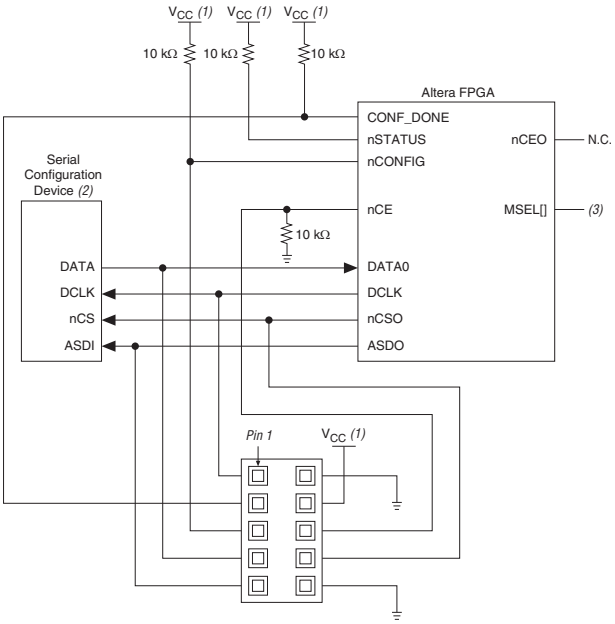
- Arria series
- Cyclone series
- all device families in the Stratix series except the Stratix device family

This section is only relevant for FPGAs that support the AS configuration scheme.

There are four signals on the serial configuration device that interface directly with the FPGA's control signals. The serial configuration device signals DATA, DCLK, ASDI, and *n*CS interface with DATA0, DCLK, ASDO, and *n*CSO control signals on the FPGA, respectively. Figure 3–2 shows a serial configuration device programmed via a download cable, which configures an FPGA in AS mode. Figure 3–3 shows a serial configuration device programmed using the APU or a third-party programmer configuring an FPGA in AS configuration mode.

For more information about the serial configuration device pin description, refer to Table 3–23.

Figure 3–2. Altera FPGA Configuration in AS Mode (Serial Configuration Device Programmed Using Download Cable) *(Note 1)*, *(4)*



Notes to Figure 3–2:

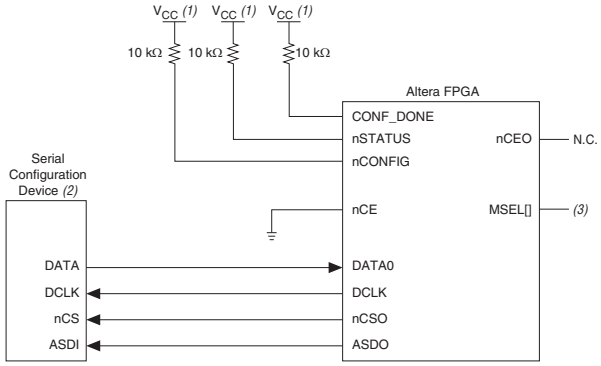
(1) For the V_{CC} value, refer to the respective FPGA family handbook Configuration chapter.

(2) Serial configuration devices cannot be cascaded.

(3) Connect the FPGA MSEL[] input pins to select the AS configuration mode. For details, refer to the respective FPGA family chapter in the *Configuration Handbook*.

(4) For more information about configuration pin I/O requirements in an AS scheme for an Altera FPGA, refer to the respective FPGA family handbook Configuration chapter.

Figure 3–3. Altera FPGA Configuration in AS Mode (Serial Configuration Device Programmed by APU or Third-Party Programmer) ***(Note 1)***, ***(4)***



Notes to Figure 3–3:

(1) For the V_{CC} value, refer to the respective FPGA family handbook Configuration chapter.

(2) Serial configuration devices cannot be cascaded.

(3) Connect the FPGA MSEL[] input pins to select the AS configuration mode. For details, refer to the respective FPGA family chapter in the *Configuration Handbook*.

(4) For more information about configuration pin I/O requirements in an AS scheme for an Altera FPGA, refer to the respective FPGA family handbook Configuration chapter.

The FPGA acts as the configuration master in the configuration flow and provides the clock to the serial configuration device. The FPGA enables the serial configuration device by pulling the *n*CS signal low via the *n*CSO signal (refer to Figure 3–2 and Figure 3–3). Subsequently, the FPGA sends the instructions and addresses to the serial configuration device via the ASDO signal. The serial configuration device responds to the instructions by sending the configuration data to the FPGA's DATA0 pin on the falling edge of DCLK. The data is latched into the FPGA on the next DCLK signal's falling edge.

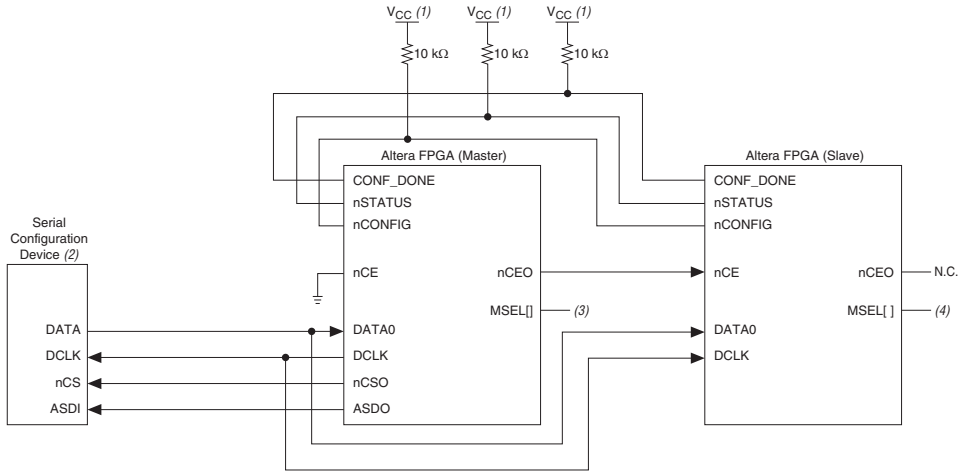
Before the FPGA enters configuration mode, ensure that V_{CC} of the EPCS is ready. If it is not, you must hold *n*CONFIG low until all power rails of EPCS are ready.

The FPGA controls the *n*STATUS and CONF_DONE pins during configuration in AS mode. If the CONF_DONE signal does not go high at the end of configuration or if the signal goes high too early, the FPGA will pulse its *n*STATUS pin low to start reconfiguration. Upon successful configuration, the FPGA releases the CONF_DONE pin, allowing the external 10-kΩ resistor to pull this signal high. Initialization begins after the CONF_DONE goes high. After initialization, the FPGA enters user mode.

For more information about configuring the FPGAs in AS mode or other configuration modes, refer to the Configuration chapter in the appropriate device handbook.

Multiple devices can be configured by a single EPCS device. However, serial configuration devices cannot be cascaded. Refer to Table 3–1 to ensure the programming file size of the cascaded FPGAs does not exceed the capacity of a serial configuration device. Figure 3–4 shows the AS configuration scheme with multiple FPGAs in the chain. The first FPGA is the configuration master and has its MSEL[] pins set to AS mode. The following FPGAs are configuration slave devices and have their MSEL[] pins set to PS mode.

Figure 3–4. Multiple Devices in AS Mode *(Note 1)*, *(5)*



Notes to Figure 3–4:

(1) For the V_{CC} value, refer to the respective FPGA family handbook Configuration chapter.

(2) Serial configuration devices cannot be cascaded.

(3) Connect the FPGA MSEL[] input pins to select the AS configuration mode. For details, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.

(4) Connect the FPGA MSEL[] input pins to select the PS configuration mode. For details, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.

(5) For more information about configuration pin I/O requirements in an AS scheme for an Altera FPGA, refer to the respective FPGA family handbook Configuration chapter.

Serial Configuration Device Memory Access

This section describes the serial configuration device's memory array organization and operation codes. Timing specifications for the memory are provided in the "Timing Information" section.

Memory Array Organization

Table 3–2 provides details about the memory array organization in EPCS128, EPCS64, EPCS16, EPCS4, and EPCS1.

Table 3–2. Memory Array Organization in Serial Configuration Devices

Details	EPCS128	EPCS64	EPCS16	EPCS4	EPCS1
Bytes (bits)	16,777,216 bytes (128 Mbits)	8,388,608 bytes (64 Mbits)	2,097,152 bytes (16 Mbits)	524,288 bytes (4 Mbits)	131,072 bytes (1 Mbit)
Number of sectors	64	128	32	8	4
Bytes (bits) per sector	262,144 bytes (2 Mbits)	65,536 bytes (512 Kbits)	65,536 bytes (512 Kbits)	65,536 bytes (512 Kbits)	32,768 bytes (256 Kbits)
Pages per sector	1,024	256	256	256	128
Total number of pages	65,536	32,768	8,192	2,048	512
Bytes per page	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes

Table 3–3 through Table 3–7 list the address range for each sector in EPCS128, EPCS64, EPCS16, EPCS4, and EPCS1.

Table 3–3. Address Range for Sectors in EPCS128 (Part 1 of 3)

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
63	H'FC0000	H'FFFFFF
62	H'F80000	H'FBFFFF
61	H'F40000	H'F7FFFF
60	H'F00000	H'F3FFFF
59	H'EC0000	H'EFFFFF
58	H'E80000	H'EBFFFF
57	H'E40000	H'E7FFFF
56	H'E00000	H'E3FFFF
55	H'DC0000	H'DFFFFF
54	H'D80000	H'DBFFFF
53	H'D40000	H'D7FFFF
52	H'D00000	H'D3FFFF
51	H'CC0000	H'CFFFFF
50	H'C80000	H'CBFFFF
49	H'C40000	H'C7FFFF
48	H'C00000	H'C3FFFF

Table 3–3. Address Range for Sectors in EPCS128 (Part 2 of 3)

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
47	H'BC0000	H'BFFFFF
46	H'B80000	H'BBFFFF
45	H'B40000	H'B7FFFF
44	H'B00000	H'B3FFFF
43	H'AC0000	H'AFFFFF
42	H'A80000	H'ABFFFF
41	H'A40000	H'A7FFFF
40	H'A00000	H'A3FFFF
39	H'9C0000	H'9FFFFF
38	H'980000	H'9BFFFF
37	H'940000	H'97FFFF
36	H'900000	H'93FFFF
35	H'8C0000	H'8FFFFF
34	H'880000	H'8BFFFF
33	H'840000	H'87FFFF
32	H'800000	H'83FFFF
31	H'7C0000	H'7FFFFF
30	H'780000	H'7BFFFF
29	H'740000	H'77FFFF
28	H'700000	H'73FFFF
27	H'6C0000	H'6FFFFF
26	H'680000	H'6BFFFF
25	H'640000	H'67FFFF
24	H'600000	H'63FFFF
23	H'5C0000	H'5FFFFF
22	H'580000	H'5BFFFF
21	H'540000	H'57FFFF
20	H'500000	H'53FFFF
19	H'4C0000	H'4FFFFF
18	H'480000	H'4BFFFF
17	H'440000	H'47FFFF
16	H'400000	H'43FFFF
15	H'3C0000	H'3FFFFF
14	H'380000	H'3BFFFF
13	H'340000	H'37FFFF
12	H'300000	H'33FFFF
11	H'2C0000	H'2FFFFF
10	H'280000	H'2BFFFF

Table 3–3. Address Range for Sectors in EPCS128 (Part 3 of 3)

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
9	H'240000	H'27FFFF
8	H'200000	H'23FFFF
7	H'1C0000	H'1FFFFF
6	H'180000	H'1BFFFF
5	H'140000	H'17FFFF
4	H'100000	H'13FFFF
3	H'0C0000	H'0FFFFF
2	H'080000	H'0BFFFF
1	H'040000	H'07FFFF
0	H'000000	H'03FFFF

Table 3–4. Address Range for Sectors in EPCS64 (Part 1 of 4)

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
127	H'7F0000	H'7FFFFF
126	H'7E0000	H'7EFFFF
125	H'7D0000	H'7DFFFF
124	H'7C0000	H'7CFFFF
123	H'7B0000	H'7BFFFF
122	H'7A0000	H'7AFFFF
121	H'790000	H'79FFFF
120	H'780000	H'78FFFF
119	H'770000	H'77FFFF
118	H'760000	H'76FFFF
117	H'750000	H'75FFFF
116	H'740000	H'74FFFF
115	H'730000	H'73FFFF
114	H'720000	H'72FFFF
113	H'710000	H'71FFFF
112	H'700000	H'70FFFF
111	H'6F0000	H'6FFFFF
110	H'6E0000	H'6EFFFF
109	H'6D0000	H'6DFFFF
108	H'6C0000	H'6CFFFF
107	H'6B0000	H'6BFFFF
106	H'6A0000	H'6AFFFF
105	H'690000	H'69FFFF
104	H'680000	H'68FFFF
103	H'670000	H'67FFFF
102	H'660000	H'66FFFF
101	H'650000	H'65FFFF
100	H'640000	H'64FFFF
99	H'630000	H'63FFFF
98	H'620000	H'62FFFF
97	H'610000	H'61FFFF
96	H'600000	H'60FFFF
95	H'5F0000	H'5FFFFF
94	H'5E0000	H'5EFFFF
93	H'5D0000	H'5DFFFF
92	H'5C0000	H'5CFFFF
91	H'5B0000	H'5BFFFF
90	H'5A0000	H'5AFFFF

Table 3–4. Address Range for Sectors in EPCS64 (Part 2 of 4)

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
89	H'590000	H'59FFFF
88	H'580000	H'58FFFF
87	H'570000	H'57FFFF
86	H'560000	H'56FFFF
85	H'550000	H'55FFFF
84	H'540000	H'54FFFF
83	H'530000	H'53FFFF
82	H'520000	H'52FFFF
81	H'510000	H'51FFFF
80	H'500000	H'50FFFF
79	H'4F0000	H'4FFFFF
78	H'4E0000	H'4EFFFF
77	H'4D0000	H'4DFFFF
76	H'4C0000	H'4CFFFF
75	H'4B0000	H'4BFFFF
74	H'4A0000	H'4AFFFF
73	H'490000	H'49FFFF
72	H'480000	H'48FFFF
71	H'470000	H'47FFFF
70	H'460000	H'46FFFF
69	H'450000	H'45FFFF
68	H'440000	H'44FFFF
67	H'430000	H'43FFFF
66	H'420000	H'42FFFF
65	H'410000	H'41FFFF
64	H'400000	H'40FFFF
63	H'3F0000	H'3FFFFF
62	H'3E0000	H'3EFFFF
61	H'3D0000	H'3DFFFF
60	H'3C0000	H'3CFFFF
59	H'3B0000	H'3BFFFF
58	H'3A0000	H'3AFFFF
57	H'390000	H'39FFFF
56	H'380000	H'38FFFF
55	H'370000	H'37FFFF
54	H'360000	H'36FFFF
53	H'350000	H'35FFFF
52	H'340000	H'34FFFF

Table 3–4. Address Range for Sectors in EPCS64 (Part 3 of 4)

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
51	H'330000	H'33FFFF
50	H'320000	H'32FFFF
49	H'310000	H'31FFFF
48	H'300000	H'30FFFF
47	H'2F0000	H'2FFFFF
46	H'2E0000	H'2EFFFF
45	H'2D0000	H'2DFFFF
44	H'2C0000	H'2CFFFF
43	H'2B0000	H'2BFFFF
42	H'2A0000	H'2AFFFF
41	H'290000	H'29FFFF
40	H'280000	H'28FFFF
39	H'270000	H'27FFFF
38	H'260000	H'26FFFF
37	H'250000	H'25FFFF
36	H'240000	H'24FFFF
35	H'230000	H'23FFFF
34	H'220000	H'22FFFF
33	H'210000	H'21FFFF
32	H'200000	H'20FFFF
31	H'1F0000	H'1FFFFF
30	H'1E0000	H'1EFFFF
29	H'1D0000	H'1DFFFF
28	H'1C0000	H'1CFFFF
27	H'1B0000	H'1BFFFF
26	H'1A0000	H'1AFFFF
25	H'190000	H'19FFFF
24	H'180000	H'18FFFF
23	H'170000	H'17FFFF
22	H'160000	H'16FFFF
21	H'150000	H'15FFFF
20	H'140000	H'14FFFF
19	H'130000	H'13FFFF
18	H'120000	H'12FFFF
17	H'110000	H'11FFFF
16	H'100000	H'10FFFF
15	H'0F0000	H'0FFFFF
14	H'0E0000	H'0EFFFF

Table 3–4. Address Range for Sectors in EPCS64 (Part 4 of 4)

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
13	H'0D0000	H'0DFFFF
12	H'0C0000	H'0CFFFF
11	H'0B0000	H'0BFFFF
10	H'0A0000	H'0AFFFF
9	H'090000	H'09FFFF
8	H'080000	H'08FFFF
7	H'070000	H'07FFFF
6	H'060000	H'06FFFF
5	H'050000	H'05FFFF
4	H'040000	H'04FFFF
3	H'030000	H'03FFFF
2	H'020000	H'02FFFF
1	H'010000	H'01FFFF
0	H'000000	H'00FFFF

Table 3–5. Address Range for Sectors in EPCS16

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
31	H'1F0000	H'1FFFFF
30	H'1E0000	H'1EFFFF
29	H'1D0000	H'1DFFFF
28	H'1C0000	H'1CFFFF
27	H'1B0000	H'1BFFFF
26	H'1A0000	H'1AFFFF
25	H'190000	H'19FFFF
24	H'180000	H'18FFFF
23	H'170000	H'17FFFF
22	H'160000	H'16FFFF
21	H'150000	H'15FFFF
20	H'140000	H'14FFFF
19	H'130000	H'13FFFF
18	H'120000	H'12FFFF
17	H'110000	H'11FFFF
16	H'100000	H'10FFFF
15	H'0F0000	H'0FFFFF
14	H'0E0000	H'0EFFFF
13	H'0D0000	H'0DFFFF
12	H'0C0000	H'0CFFFF
11	H'0B0000	H'0BFFFF
10	H'0A0000	H'0AFFFF
9	H'090000	H'09FFFF
8	H'080000	H'08FFFF
7	H'070000	H'07FFFF
6	H'060000	H'06FFFF
5	H'050000	H'05FFFF
4	H'040000	H'04FFFF
3	H'030000	H'03FFFF
2	H'020000	H'02FFFF
1	H'010000	H'01FFFF
0	H'000000	H'00FFFF

Table 3–6. Address Range for Sectors in EPCS4

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
7	H'70000	H'7FFFF
6	H'60000	H'6FFFF
5	H'50000	H'5FFFF
4	H'40000	H'4FFFF
3	H'30000	H'3FFFF
2	H'20000	H'2FFFF
1	H'10000	H'1FFFF
0	H'00000	H'0FFFF

Table 3–7. Address Range for Sectors in EPCS1

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
3	H'18000	H'1FFFF
2	H'10000	H'17FFF
1	H'08000	H'0FFFF
0	H'00000	H'07FFF

Operation Codes

This section describes the operations that can be used to access the memory in serial configuration devices. The DATA, DCLK, ASDI, and *n*CS signals access the memory in serial configuration devices. All serial configuration device operation codes, addresses and data are shifted in and out of the device serially, with the most significant bit (MSB) first.

The device samples the active serial data input on the first rising edge of the DCLK after the active low chip select (*n*CS) input signal is driven low. Shift the operation code (MSB first) serially into the serial configuration device through the active serial data input (ASDI) pin. Each operation code bit is latched into the serial configuration device on the rising edge of the DCLK.

Different operations require a different sequence of inputs. While executing an operation, you must shift in the desired operation code, followed by the address bytes, data bytes, both, or neither. The device must drive *n*CS high after the last bit of the operation sequence is shifted in. Table 3–8 lists the operation sequence for every operation supported by the serial configuration devices.

For the read byte, read status, and read silicon ID operations, the shifted-in operation sequence is followed by data shifted out on the DATA pin. You can drive the *n*CS pin high after any bit of the data-out sequence is shifted out.

For the write byte, erase bulk, erase sector, write enable, write disable, and write status operations, drive the nCS pin high exactly at a byte boundary (drive the nCS pin high a multiple of eight clock pulses after the nCS pin is driven low); otherwise, the operation is rejected and is not executed.

All attempts to access the memory contents while a write or erase cycle is in progress will not be granted, and the write or erase cycle will continue unaffected.

Table 3–8. Operation Codes for Serial Configuration Devices

Operation	Operation Code *(1)*	Address Bytes	Dummy Bytes	Data Bytes	DCLK f_{MAX} (MHz)
Write enable	0000 0110	0	0	0	25
Write disable	0000 0100	0	0	0	25
Read status	0000 0101	0	0	1 to infinite *(2)*	25
Read bytes	0000 0011	3	0	1 to infinite *(2)*	20
Read silicon ID *(4)*	1010 1011	0	3	1 to infinite *(2)*	25
Fast read	0000 1011	3	1	1 to infinite *(2)*	40
Write status	0000 0001	0	0	1	25
Write bytes	0000 0010	3	0	1 to 256 *(3)*	25
Erase bulk	1100 0111	0	0	0	25
Erase sector	1101 1000	3	0	0	25
Read Device Identification *(5)*	1001 1111	0	2	1 to infinite *(2)*	25

Notes to Table 3–8:

(1) The MSB is listed first and the least significant bit (LSB) is listed last.

(2) The status register, data or silicon ID are read out at least once on the DATA pin and will continuously be read out until nCS is driven high.

(3) Write bytes operation requires at least one data byte on the DATA pin. If more than 256 bytes are sent to the device, only the last 256 bytes are written to the memory.

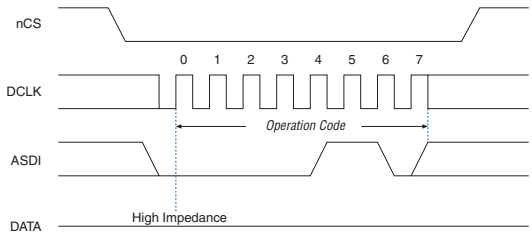
(4) Read silicon ID operation is available only for EPCS1, EPCS4, EPCS16, and EPCS64.

(5) Read Device Identification operation is available only for EPCS128.

Write Enable Operation

The write enable operation code is b'0000 0110, and the MSB is listed first. The write enable operation sets the write enable latch bit, which is bit 1 in the status register. Always set the write enable latch bit before write bytes, write status, erase bulk, and erase sector operations. Figure 3–5 shows the timing diagram for the write enable operation. Figure 3–7 and Figure 3–8 show the status register bit definitions.

Figure 3–5. Write Enable Operation Timing Diagram



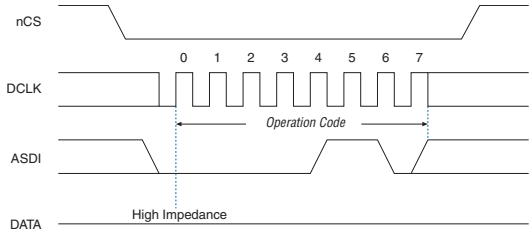
Write Disable Operation

The write disable operation code is `b'0000 0100`, with the MSB listed first. The write disable operation resets the write enable latch bit, which is bit `1` in the status register. To prevent the memory from being written unintentionally, the write enable latch bit is automatically reset when implementing the write disable operation as well as under the following conditions:

- Power up
- Write bytes operation completion
- Write status operation completion
- Erase bulk operation completion
- Erase sector operation completion

Figure 3–6 shows the timing diagram for the write disable operation.

Figure 3–6. Write Disable Operation Timing Diagram



Read Status Operation

The read status operation code is `b'0000 0101`, with the MSB listed first. You can use the read status operation to read the status register. Figure 3–7 and Figure 3–8 show the status bits in the status register of the serial configuration devices.

Figure 3–7. EPCS4, EPCS16, EPCS64, and EPCS128 Status Register Status Bits

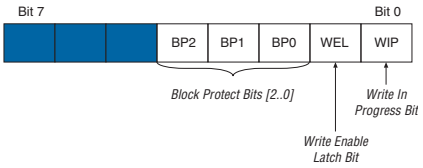
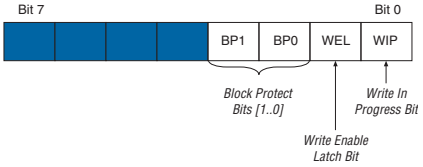


Figure 3–8. EPCS1 Status Register Status Bits



Setting the write in progress bit to `1` indicates that the serial configuration device is busy with a write or erase cycle. Resetting the write in progress bit to `0` means no write or erase cycle is in progress.

Resetting the write enable latch bit to 0 indicates that no write or erase cycle will be accepted. Set the write enable latch bit to 1 before every write bytes, write status, erase bulk, and erase sector operation.

The non-volatile block protect bits determine the area of the memory protected from being written or erased unintentionally. Table 3–9 through Table 3–13 list the protected area in the serial configuration devices with reference to the block protect bits. The erase bulk operation is only available when all the block protect bits are 0. When any of the block protect bits are set to 1, the relevant area is protected from being written by write bytes operations or erased by erase sector operations.

Table 3–9. Block Protection Bits in EPCS1

Status Register Content		Memory Content	
BP1 Bit	**BP0 Bit**	**Protected Area**	**Unprotected Area**
0	0	None	All four sectors: 0 to 3
0	1	Sector 3	Three sectors: 0 to 2
1	0	Two sectors: 2 and 3	Two sectors: 0 and 1
1	1	All sectors	None

Table 3–10. Block Protection Bits in EPCS4

Status Register Content			Memory Content	
BP2 Bit	**BP1 Bit**	**BP0 Bit**	**Protected Area**	**Unprotected Area**
0	0	0	None	All eight sectors: 0 to 7
0	0	1	Sector 7	Seven sectors: 0 to 6
0	1	0	Sectors 6 and 7	Six sectors: 0 to 5
0	1	1	Four sectors: 4 to 7	Four sectors: 0 to 3
1	0	0	All sectors	None
1	0	1	All sectors	None
1	1	0	All sectors	None
1	1	1	All sectors	None

Table 3–11. Block Protection Bits in EPCS16 (Part 1 of 2)

Status Register Content			Memory Content	
BP2 Bit	**BP1 Bit**	**BP0 Bit**	**Protected Area**	**Unprotected Area**
0	0	0	None	All sectors (32 sectors 0 to 31)
0	0	1	Upper 32nd (Sector 31)	Lower 31/32nds (31 sectors: 0 to 30)
0	1	0	Upper sixteenth (two sectors: 30 and 31)	Lower 15/16ths (30 sectors: 0 to 29)
0	1	1	Upper eighth (four sectors: 28 to 31)	Lower seven-eighths (28 sectors: 0 to 27)
1	0	0	Upper quarter (eight sectors: 24 to 31)	Lower three-quarters (24 sectors: 0 to 23)
1	0	1	Upper half (sixteen sectors: 16 to 31)	Lower half (16 sectors: 0 to 15)

Table 3–11. Block Protection Bits in EPCS16 (Part 2 of 2)

Status Register Content			Memory Content	
BP2 Bit	**BP1 Bit**	**BP0 Bit**	**Protected Area**	**Unprotected Area**
1	1	0	All sectors (32 sectors: 0 to 31)	None
1	1	1	All sectors (32 sectors: 0 to 31)	None

Table 3–12. Block Protection Bits in EPCS64

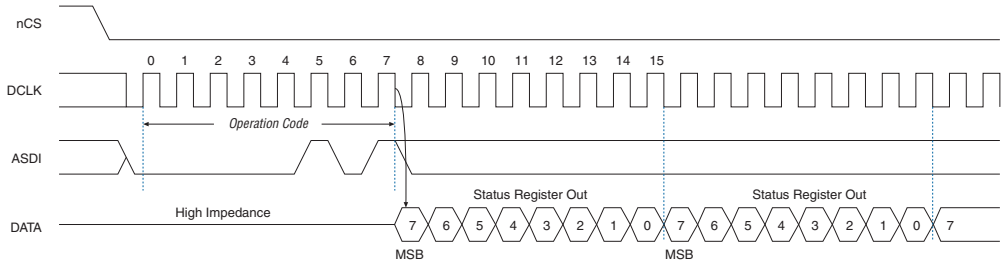
Status Register Content			Memory Content	
BP2 Bit	**BP1 Bit**	**BP0 Bit**	**Protected Area**	**Unprotected Area**
0	0	0	None	All sectors (128 sectors: 0 to 127)
0	0	1	Upper 64th (2 sectors: 126 and 127)	Lower 63/64ths (126 sectors: 0 to 125)
0	1	0	Upper 32nd (4 sectors: 124 to 127)	Lower 31/32nds (124 sectors: 0 to 123)
0	1	1	Upper sixteenth (8 sectors: 120 to 127)	Lower 15/16ths (120 sectors: 0 to 119)
1	0	0	Upper eighth (16 sectors: 112 to 127)	Lower seven-eighths (112 sectors: 0 to 111)
1	0	1	Upper quarter (32 sectors: 96 to 127)	Lower three-quarters (96 sectors: 0 to 95)
1	1	0	Upper half (64 sectors: 64 to 127)	Lower half (64 sectors: 0 to 63)
1	1	1	All sectors (128 sectors: 0 to 127)	None

Table 3–13. Block Protection Bits in EPCS128

Status Register Content			Memory Content	
BP2 Bit	**BP1 Bit**	**BP0 Bit**	**Protected Area**	**Unprotected Area**
0	0	0	None	All sectors (64 sectors: 0 to 63)
0	0	1	Upper 64th (1 sector: 63)	Lower 63/64ths (63 sectors: 0 to 62)
0	1	0	Upper 32nd (2 sectors: 62 to 63)	Lower 31/32nds (62 sectors: 0 to 61)
0	1	1	Upper 16th (4 sectors: 60 to 63)	Lower 15/16ths (60 sectors: 0 to 59)
1	0	0	Upper 8th (8 sectors: 56 to 63)	Lower seven-eighths (56 sectors: 0 to 55)
1	0	1	Upper quarter (16 sectors: 48 to 63)	Lower three-quarters (48 sectors: 0 to 47)
1	1	0	Upper half (32 sectors: 32 to 63)	Lower half (32 sectors: 0 to 31)
1	1	1	All sectors (64 sectors: 0 to 63)	None

You can read the status register at any time, even while a write or erase cycle is in progress. When one of these cycles is in progress, you can check the write in progress bit (bit 0 of the status register) before sending a new operation to the device. The device can also read the status register continuously, as shown in Figure 3–9.

Figure 3–9. Read Status Operation Timing Diagram


nCS
DCLK
ASDI
DATA
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Operation Code
High Impedance
Status Register Out
Status Register Out
7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7
MSB
MSB

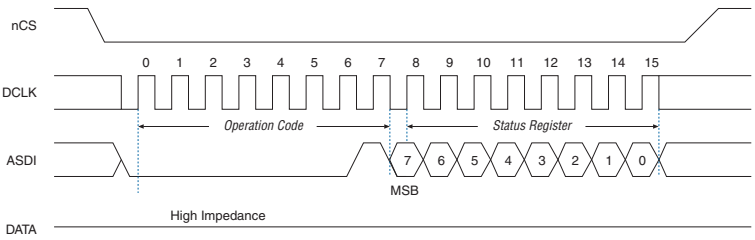
Write Status Operation

The write status operation code is b'0000 0001, with the MSB listed first. Use the write status operation to set the status register block protection bits. The write status operation has no effect on the other bits. Therefore, you can implement this operation to protect certain memory sectors, as defined in Table 3–9 through Table 3–13. After setting the block protect bits, the protected memory sectors are treated as read-only memory. You must execute the write enable operation before the write status operation so the device sets the status register's write enable latch bit to 1.

The write status operation is implemented by driving *n*CS low, followed by shifting in the write status operation code and one data byte for the status register on the ASDI pin. Figure 3–10 shows the timing diagram for the write status operation. *n*CS must be driven high after the eighth bit of the data byte has been latched in, otherwise, the write status operation is not executed.

Immediately after *n*CS drives high, the device initiates the self-timed write status cycle. The self-timed write status cycle usually takes 5 ms for all serial configuration devices and is guaranteed to be less than 15 ms (refer to t_{WS} in Table 3–16). You must account for this delay to ensure that the status register is written with desired block protect bits. Alternatively, you can check the write in progress bit in the status register by executing the read status operation while the self-timed write status cycle is in progress. The write in progress bit is 1 during the self-timed write status cycle, and 0 when it is complete.

Figure 3–10. Write Status Operation Timing Diagram

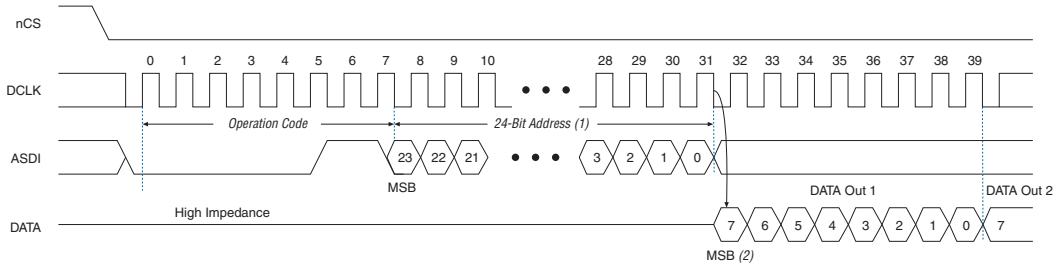


Read Bytes Operation

The read bytes operation code is b'0000 0011, with the MSB listed first. To read the memory contents of the serial configuration device, the device is first selected by driving *n*CS low. Then, the read bytes operation code is shifted in followed by a 3-byte address (A[23..0]). Each address bit must be latched in on the rising edge of the DCLK. After the address is latched in, the memory contents of the specified address are shifted out serially on the DATA pin, beginning with the MSB. For reading Raw Programming Data files (**.rpd**), the content is shifted out serially beginning with the LSB. Each data bit is shifted out on the falling edge of DCLK. The maximum DCLK frequency during the read bytes operation is 20 MHz. Figure 3–11 shows the timing diagram for the read bytes operation.

The first byte address can be at any location. The device automatically increments the address to the next higher address after shifting out each byte of data. Therefore, the device can read the whole memory with a single read bytes operation. When the device reaches the highest address, the address counter restarts at 0x000000, allowing the memory contents to be read out indefinitely until the read bytes operation is terminated by driving *n*CS high. The device can drive *n*CS high any time after data is shifted out. If the read bytes operation is shifted in while a write or erase cycle is in progress, the operation is not executed and has no effect on the write or erase cycle in progress.

Figure 3–11. Read Bytes Operation Timing Diagram



Notes to Figure 3–11:

(1) Address bit A[23] is a don't-care bit in EPCS64. Address bits A[23..21] are don't-care bits in EPCS16. Address bits A[23..19] are don't-care bits in EPCS4. Address bits A[23..17] are don't-care bits in the EPCS1.

(2) For **.rpd** files, the read sequence shifts out the LSB of the data byte first.

Fast Read Operation

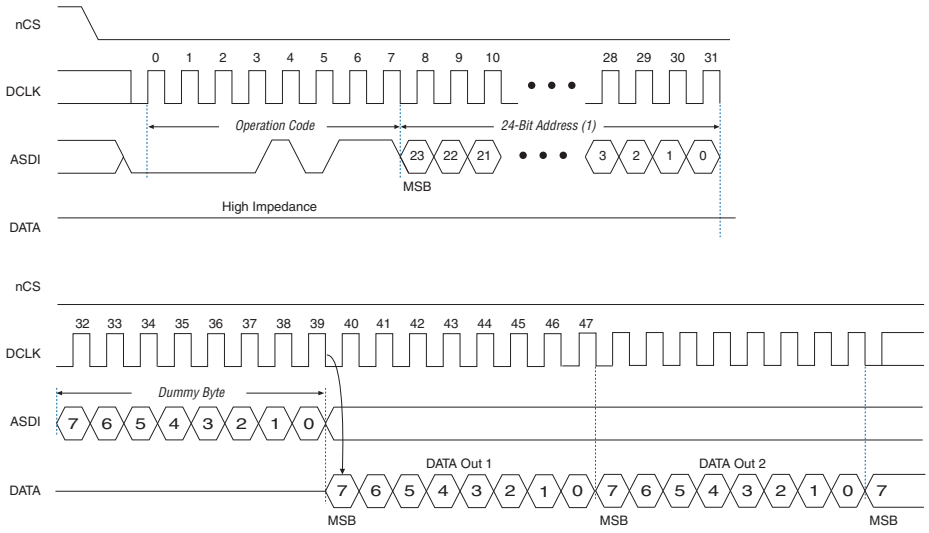
The device is first selected by driving *n*CS low. The fast read instruction code is followed by a 3-byte address (A23-A0) and a dummy byte, each bit being latched-in during the rising edge of DCLK. Then the memory contents, at that address, is shifted out on DATA, each bit being shifted out, at a maximum frequency of 40 MHz, during the falling edge of DCLK.

The instruction sequence is shown in Figure 3–12.

The first byte addressed can be at any location. The address is automatically incremented to the next higher address after each byte of data is shifted out. The whole memory can, therefore, be read with a single fast read instruction. When the highest address is reached, the address counter rolls over to 000000h, allowing the read sequence to be continued indefinitely.

The fast read instruction is terminated by driving nCS high at any time during data output. Any fast read instruction is rejected during the Erase, Program, or Write operations without any effect on the operation that is in progress.

Figure 3–12. FAST_READ Operation Timing Diagram



Note to Figure 3–12:

(1) Address bit A[23] is a don't-care bit in EPCS64. Address bits A[23..21] are don't-care bits in EPCS16. Address bits A[23..19] are don't-care bits in EPCS4. Address bits A[23..17] are don't-care bits in the EPCS1.

Read Silicon ID Operation

The read silicon ID operation code is b'1010 1011, with the MSB listed first. Only EPCS1, EPCS4, EPCS16, and EPCS64 support this operation. It reads the serial configuration device's 8-bit silicon ID from the DATA output pin. If this operation is shifted in during an erase or write cycle, it is ignored and has no effect on the cycle that is in progress.

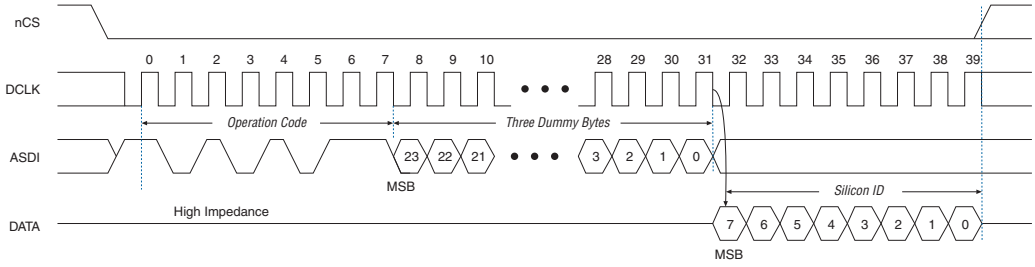
Table 3–14 lists the serial configuration device silicon IDs.

Table 3–14. Serial Configuration Device Silicon ID

Serial Configuration Device	Silicon ID (Binary Value)
EPCS1	b'0001 0000
EPCS4	b'0001 0010
EPCS16	b'0001 0100
EPCS64	b'0001 0110

The device implements the read silicon ID operation by driving nCS low then shifting in the read silicon ID operation code followed by three dummy bytes on ASDI. The serial configuration device's 8-bit silicon ID is then shifted out on the DATA pin on the falling edge of DCLK, as shown in Figure 3–13. The device can terminate the read silicon ID operation by driving nCS high after the silicon ID has been read at least once. Sending additional clock cycles on DCLK while nCS is driven low can cause the silicon ID to be shifted out repeatedly.

Figure 3–13. Read Silicon ID Operation Timing Diagram ***(Note 1)***



Note to Figure 3–13:

(1) Only EPCS1, EPCS4, EPCS16, and EPCS64 support Read Silicon ID operation.

Read Device Identification Operation

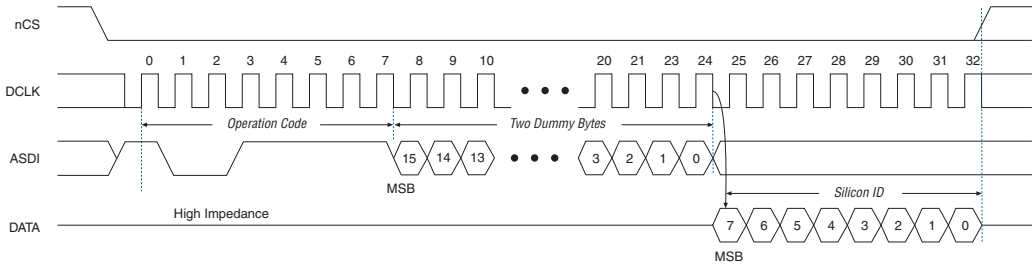
The read device identification operation code is b'1001 1111, with the MSB listed first. Only EPCS128 supports this operation. It reads the serial configuration device's 8-bit device identification from the DATA output pin. If this operation is shifted in during an erase or write cycle, it is ignored and has no effect on the cycle that is in progress. Table 3–15 shows the serial configuration device identification.

Table 3–15. Serial Configuration Device Identification

Serial Configuration Device	Silicon ID (Binary Value)
EPCS128	b'0001 1000

The device implements the read device identification operation by driving nCS low then shifting in the read device identification operation code followed by two dummy byte on ASDI. The serial configuration device's 16-bit device identification is then shifted out on the DATA pin on the falling edge of DCLK, as shown in Figure 3–14. The device can terminate the read device identification operation by driving nCS high after reading the device identification at least once.

Figure 3–14. Read Device Identification Operation Timing Diagram *(Note 1)*



Note to Figure 3–14:

(1) Only EPCS128 supports read device identification operation.

Write Bytes Operation

The write bytes operation code is b'0000 0010, with the MSB listed first. The write bytes operation allows bytes to be written to the memory. The write enable operation must be executed prior to the write bytes operation to set the write enable latch bit in the status register to 1.

The write bytes operation is implemented by driving *n*CS low, followed by the write bytes operation code, three address bytes and a minimum one data byte on ASDI. If the eight least significant address bits (A[7..0]) are not all 0, all sent data that goes beyond the end of the current page is not written into the next page. Instead, this data is written at the start address of the same page (from the address whose eight LSBs are all 0). Drive *n*CS low during the entire write bytes operation sequence, as shown in Figure 3–15.

If more than 256 data bytes are shifted into the serial configuration device with a write bytes operation, the previously latched data is discarded and the last 256 bytes are written to the page. However, if less than 256 data bytes are shifted into the serial configuration device, they are guaranteed to be written at the specified addresses and the other bytes of the same page are unaffected.

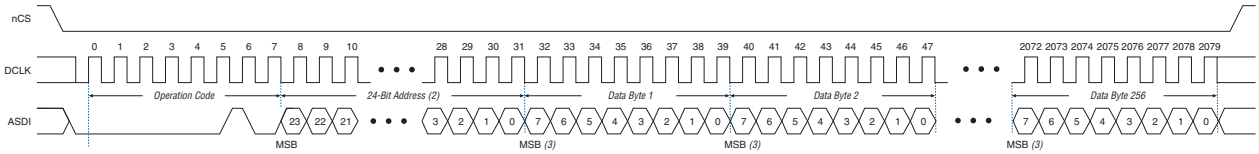
If the design must write more than 256 data bytes to the memory, it needs more than one page of memory. Send the write enable and write bytes operation codes followed by three new targeted address bytes and 256 data bytes before a new page is written.

*n*CS must be driven high after the eighth bit of the last data byte has been latched in. Otherwise, the device will not execute the write bytes operation. The write enable latch bit in the status register is reset to 0 before the completion of each write bytes operation. Therefore, the write enable operation must be carried out before the next write bytes operation.

The device initiates the self-timed write cycle immediately after *n*CS is driven high. Refer to t_{WB} in Table 3–16 on page 3–27 for the self-timed write cycle time for the respective EPCS devices. Therefore, you must account for this amount of delay before another page of memory is written. Alternatively, you can check the status register's write in progress bit by executing the read status operation while the self-timed write cycle is in progress. The write in progress bit is set to 1 during the self-timed write cycle, and 0 when it is complete.

The bytes of serial configuration devices memory must be erased to all `1` or `0xFF` before write bytes operation is implemented. This can be achieved by either using the erase sector instruction in a sector, or the erase bulk instruction throughout the entire memory.

Figure 3–15. Write Bytes Operation Timing Diagram ***(Note 1)***



Notes to Figure 3–15:

(1) Use the erase sector or the erase bulk instruction to initialize the memory bytes of the serial configuration devices to all `1` or `0xFF` before implementing the write bytes operation.

(2) Address bit `A[23]` is a don't-care bit in EPCS64. Address bits `A[23..21]` are don't-care bits in EPCS16. Address bits `A[23..19]` are don't-care bits in EPCS4. Address bits `A[23..17]` are don't-care bits in EPCS1.

(3) For **.rpd** files, write the LSB of the data byte first.

Erase Bulk Operation

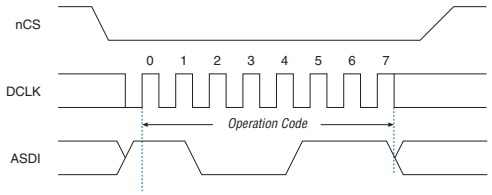
The erase bulk operation code is `b'1100 0111`, with the MSB listed first. The erase bulk operation sets all memory bits to `1` or `0xFF`. Similar to the write bytes operation, the write enable operation must be executed prior to the erase bulk operation so that the write enable latch bit in the status register is set to `1`.

You can implement the erase bulk operation by driving *n*`CS` low and then shifting in the erase bulk operation code on the `ASDI` pin. *n*`CS` must be driven high after the eighth bit of the erase bulk operation code has been latched in. Figure 3–16 shows the timing diagram.

The device initiates the self-timed erase bulk cycle immediately after *n*`CS` is driven high. Refer to t_{EB} in Table 3–16 for the self-timed erase bulk cycle time for the respective EPCS devices.

You must account for this delay before accessing the memory contents. Alternatively, you can check the write in progress bit in the status register by executing the read status operation while the self-timed erase cycle is in progress. The write in progress bit is `1` during the self-timed erase cycle and `0` when it is complete. The write enable latch bit in the status register is reset to `0` before the erase cycle is complete.

Figure 3–16. Erase Bulk Operation Timing Diagram



Erase Sector Operation

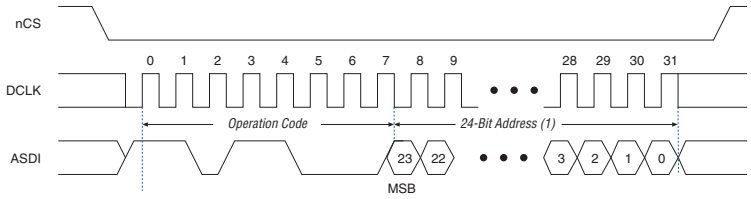
The erase sector operation code is b'1101 1000, with the MSB listed first. The erase sector operation allows the user to erase a certain sector in the serial configuration device by setting all bits inside the sector to 1 or 0xFF. This operation is useful for users who access the unused sectors as general purpose memory in their applications.

The write enable operation must be executed prior to the erase sector operation so that the write enable latch bit in the status register is set to 1.

The erase sector operation is implemented by first driving *n*CS low, then shifting in the erase sector operation code and the three address bytes of the chosen sector on the ASDI pin. The three address bytes for the erase sector operation can be any address inside the specified sector. (Refer to Table 3–3 through Table 3–7 for sector address range information.) Drive *n*CS high after the eighth bit of the erase sector operation code has been latched in. Figure 3–17 shows the timing diagram.

Immediately after the device drives *n*CS high, the self-timed erase sector cycle is initiated. Refer to t_{ES} in Table 3–16 for the self-timed erase sector cycle time for the respective EPCS devices. You must account for this amount of delay before the memory contents can be accessed. Alternatively, you can check the write in progress bit in the status register by executing the read status operation while the erase cycle is in progress. The write in progress bit is 1 during the self-timed erase cycle and 0 when it is complete. The write enable latch bit in the status register resets to 0 before the erase cycle is complete.

Figure 3–17. Erase Sector Operation Timing Diagram



Note to Figure 3–17:

(1) Address bit A[23] is a don't-care bit in EPCS64. Address bits A[23..21] are don't-care bits in EPCS16. Address bits A[23..19] are don't-care bits in EPCS4. Address bits A[23..17] are don't-care bits in EPCS1.

Power and Operation

This section describes the power modes, power-on reset (POR) delay, error detection, and initial programming state of serial configuration devices.

Power Mode

Serial configuration devices support active power and standby power modes. When *n*CS is low, the device is enabled and is in active power mode. The FPGA is configured while in active power mode. When *n*CS is high, the device is disabled but could remain in active power mode until all internal cycles have completed (such as write or erase operations). The serial configuration device then goes into stand-by power mode. The I_{CC1} parameter specifies the V_{CC} supply current when the device is in active power mode and the I_{CC0} parameter specifies the current when the device is in stand-by power mode (refer to Table 3–21).

Power-On Reset

During initial power-up, a POR delay occurs to ensure the system voltage levels have stabilized. During AS configuration, the FPGA controls the configuration and has a longer POR delay than the serial configuration device.

For the POR delay time, refer to the configuration chapter in the appropriate device handbook.

Error Detection

During AS configuration with the serial configuration device, the FPGA monitors the configuration status through the *n*STATUS and CONF_DONE pins. If an error condition occurs (*n*STATUS drives low) or if the CONF_DONE pin does not go high, the FPGA will begin reconfiguration by pulsing the *n*STATUS and *n*CSO signals, which controls the chip select pin on the serial configuration device (*n*CS).

After an error, configuration automatically restarts if the **Auto-Restart Upon Frame Error** option is turned on in the Quartus® II software. If the option is turned off, the system must monitor the *n*STATUS signal for errors and then pulse the *n*CONFIG signal low to restart configuration.

Timing Information

Figure 3–18 shows the timing waveform for write operation to the serial configuration device.

Figure 3–18. Write Operation Timing

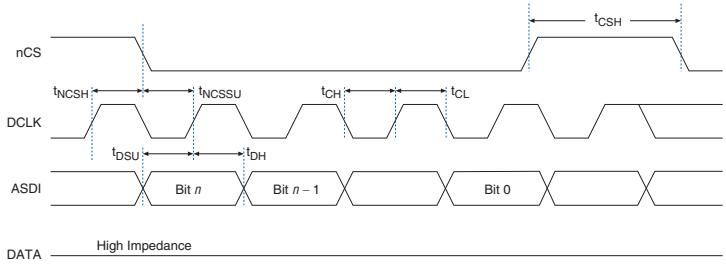


Table 3–16 defines the serial configuration device timing parameters for write operation.

Table 3–16. Write Operation Parameters (Part 1 of 2)

Symbol	Parameter	Min	Typ	Max	Unit
f_{WCLK}	Write clock frequency (from FPGA, download cable, or embedded processor) for write enable, write disable, read status, read silicon ID, write bytes, erase bulk, and erase sector operations	—	—	25	MHz
t_{CH}	DCLK high time	20	—	—	ns
t_{CL}	DCLK low time	20	—	—	ns
t_{NCSSU}	Chip select (*n*CS) setup time	10	—	—	ns

Table 3–16. Write Operation Parameters (Part 2 of 2)

Symbol	Parameter	Min	Typ	Max	Unit
t_{NCSH}	Chip select (nCS) hold time	10	—	—	ns
t_{DSU}	Data (ASDI) in setup time before rising edge on DCLK	5	—	—	ns
t_{DH}	Data (ASDI) hold time after rising edge on DCLK	5	—	—	ns
t_{CSH}	Chip select high time	100	—	—	ns
t_{WB} *(1)*	Write bytes cycle time for EPCS1, EPCS4, EPCS16, and EPCS64	—	1.5	5	ms
	Write bytes cycle time for EPCS128	—	2.5	7	ms
t_{WS} *(1)*	Write status cycle time	—	5	15	ms
t_{EB} *(1)*	Erase bulk cycle time for EPCS1	—	3	6	s
	Erase bulk cycle time for EPCS4	—	5	10	s
	Erase bulk cycle time for EPCS16	—	17	40	s
	Erase bulk cycle time for EPCS64	—	68	160	s
	Erase bulk cycle time for EPCS128	—	105	250	s
t_{ES} *(1)*	Erase sector cycle time for EPCS1, EPCS4, EPCS16, and EPCS64	—	2	3	s
	Erase sector cycle time for EPCS128	—	2	6	s

Note to Table 3–16:

(1) These parameters are not shown in Figure 3–18.

Figure 3–19 shows the timing waveform for the serial configuration device's read operation.

Figure 3–19. Read Operation Timing

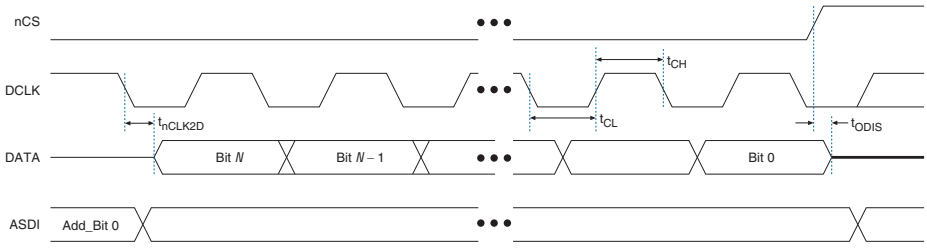


Table 3–17 defines the serial configuration device timing parameters for read operation.

Table 3–17. Read Operation Parameters

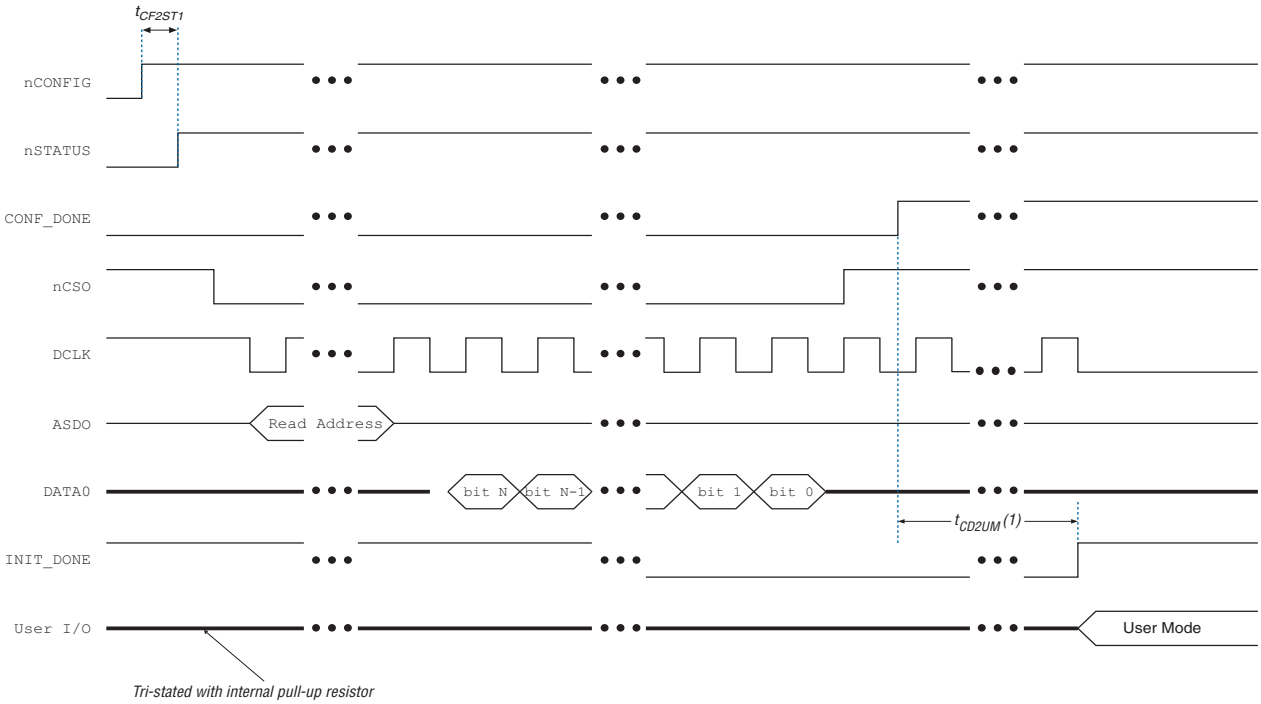
Symbol	Parameter	Min	Max	Unit
f_{RCLK}	Read clock frequency (from FPGA or embedded processor) for read bytes operation	—	20	MHz
t_{CH}	DCLK high time	25	—	ns
t_{CL}	DCLK low time	25	—	ns
t_{ODIS}	Output disable time after read	—	15	ns
t_{nCLK2D}	Clock falling edge to data	—	8	ns

Existing batches of EPCS1 and EPCS4 manufactured on 0.15 µm process geometry support AS configuration up to 40 MHz. However, batches of EPCS1 and EPCS4 manufactured on 0.18 µm process geometry support only up to 20 MHz. EPCS16, EPCS64, and EPCS128 are not affected.

For information about product traceability and transition date to differentiate between 0.15 µm process geometry and 0.18 µm process geometry EPCS1 and EPCS4, refer to the Process Change Notification *PCN 0514: Manufacturing Changes on EPCS Family*.

Figure 3–20 shows the timing waveform for FPGA AS configuration scheme using a serial configuration device.

Figure 3–20. AS Configuration Timing



Note to Figure 3–20:

(1) t_{CD2UM} is a FPGA dependent parameter. For more information, refer to the respective device configuration chapters.

For more information about the timing parameters in Figure 3–20, refer to the respective FPGA family handbook Configuration chapter.

Programming and Configuration File Support

The Quartus II software provides programming support for serial configuration devices. After selecting the serial configuration device, the Quartus II software automatically generates the Programmer Object File (**.pof**) to program the device. The software allows users to select the appropriate serial configuration device density that most efficiently stores the configuration data for a selected FPGA.

The serial configuration device can be programmed in-system by an external microprocessor using SRunner. SRunner is a software driver developed for embedded serial configuration device programming that you can customize to fit in different embedded systems. The SRunner can read **.rpd** files and write to the serial configuration devices. The programming time is comparable to the Quartus II software programming time. Note that writing and reading the **.rpd** file to the EPCS is different from other data and address bytes. The LSB of **.rpd** bytes must be shifted out first during the read bytes instruction and the LSB of **.rpd** bytes must be shifted in first during the write bytes instruction. This is because the FPGA reads the LSB of the **.rpd** data first during the configuration process.

For more information about SRunner, refer to *AN 418: SRunner: An Embedded Solution for Serial Configuration Device Programming User Guide* and the source code on the Altera website (www.altera.com).

Serial configuration devices can be programmed using the APU with the appropriate programming adapter (PLMSEPC-8) via the Quartus II software, USB Blaster, EthernetBlaster, or the ByteBlaster II download cable via the Quartus II software. In addition, many third-party programmers, such as BP Microsystems and System General, offer programming hardware that supports serial configuration devices.

During in-system programming of a serial configuration device via the USB Blaster, EthernetBlaster, or ByteBlaster II download cable, the cable pulls *n*CONFIG low to reset the FPGA and overrides the 10-kΩ pull-down resistor on the FPGA's *n*CE pin (refer to Figure 3–2). The download cable then uses the four interface pins (DATA, *n*CS, ASDI, and DCLK) to program the serial configuration device. When the programming is complete, the download cable releases the serial configuration device's four interface pins and the FPGA's *n*CE pin, and pulses *n*CONFIG to start configuration.

The FPGA can program the serial configuration device in-system using the JTAG interface with the Serial FlashLoader. This solution allows you to indirectly program the serial configuration device using the same JTAG interface that is used to configure the FPGA.

For more information about the Serial FlashLoader, refer to *AN 370: Using the Serial FlashLoader with the Quartus II Software*.

For more information about programming and configuration support, refer to the following documents:

- *Altera Programming Hardware Data Sheet*
- *Programming Hardware Manufacturers*
- *USB-Blaster Download Cable User Guide*
- *ByteBlaster II Download Cable User Guide*

- *EthernetBlaster Communications Cable User Guide*

Operating Conditions

Table 3–18 through Table 3–22 provide information about absolute maximum ratings, recommended operating conditions, DC operating conditions, and capacitance for serial configuration devices.

Table 3–18. Absolute Maximum Ratings *(Note 1)*

Symbol	Parameter	Condition	Min	Max	Unit
V_{CC}	Supply voltage for EPCS1, EPCS4, and EPCS16	With respect to ground	–0.6	4.0	V
	Supply voltage for EPCS64 and EPCS128	With respect to ground	–0.2	4.0	V
V_I	DC input voltage for EPCS1, EPCS4, and EPCS16	With respect to ground	–0.6	4.0	V
	DC input voltage for EPCS64 and EPCS128	With respect to ground	–0.5	4.0	V
I_{MAX}	DC V_{CC} or GND current	—	—	15	mA
I_{OUT}	DC output current per pin	—	–25	25	mA
P_D	Power dissipation	—	—	54	mW
T_{STG}	Storage temperature	No bias	–65	150	°C
T_{AMB}	Ambient temperature	Under bias	–65	135	°C
T_J	Junction temperature	Under bias	—	135	°C

Table 3–19. Recommended Operating Conditions

Symbol	Parameter	Conditions	Min	Max	Unit
V_{CC}	Supply voltage	*(2)*	2.7	3.6	V
V_I	Input voltage	Respect to GND	–0.3	0.3 + V_{CC}	V
V_O	Output voltage	—	0	V_{CC}	V
T_A	Operating temperature	For commercial use	0	70	°C
		For industrial use	–40	85	°C
		For extended industrial temperature	-40	125	°C
		For military use (For EPCS4, EPCS16, EPCS64, and EPCS128 devices only)	–55	125	°C
t_R	Input rise time	—	—	5	ns
t_F	Input fall time	—	—	5	ns

Table 3–20. DC Operating Conditions (Part 1 of 2)

Symbol	Parameter	Conditions	Min	Max	Unit
V_{IH}	High-level input voltage for EPCS1, EPCS4, and EPCS16	—	0.6 x V_{CC}	V_{CC} + 0.4	V
	High-level input voltage for EPCS64 and EPCS128	—	0.6 x V_{CC}	V_{CC} + 0.2	V

Table 3–20. DC Operating Conditions (Part 2 of 2)

Symbol	Parameter	Conditions	Min	Max	Unit
V_{IL}	Low-level input voltage	—	–0.5	0.3 x V_{CC}	V
V_{OH}	High-level output voltage	I_{OH} = –100 µA *(3)*	V_{CC} – 0.2	—	V
V_{OL}	Low-level output voltage	I_{OL} = 1.6 mA *(3)*	—	0.4	V
I_I	Input leakage current	V_I = V_{CC} or GND	–10	10	µA
I_{OZ}	Tri-state output off-state current	V_O = V_{CC} or GND	–10	10	µA

Table 3–21. I_{CC} Supply Current

Symbol	Parameter	Conditions	Min	Max	Unit
I_{CC0}	V_{CC} supply current (standby) for EPCS1, EPCS4, and EPCS16	—	—	50	µA
	V_{CC} supply current (standby) for EPCS64 and EPCS128	—	—	100	µA
I_{CC1}	V_{CC} supply current (during active power mode) for EPCS1, EPCS4, and EPCS16	—	5	15	mA
	V_{CC} supply current (during active power mode) for EPCS64 and EPCS128	—	5	20	mA

Table 3–22. Capacitance *(Note 4)*

Symbol	Parameter	Conditions	Min	Max	Unit
C_{IN}	Input pin capacitance	V_{IN} = 0 V	—	6	pF
C_{OUT}	Output pin capacitance	V_{OUT} = 0 V	—	8	pF

Notes to Table 3–18 through Table 3–22:

(1) For more information, refer to the *Operating Requirements for Altera Devices Data Sheet*.
(2) Maximum V_{CC} rise time is 100 ms.
(3) The I_{OH} parameter refers to high-level TTL or CMOS output current; the I_{OL} parameter refers to low-level TTL or CMOS output current.
(4) Capacitance is sample-tested only at T_A = 25 ×C and at a 20-MHz frequency.

Pin Information

As shown in Figure 3–21 and Figure 3–22, the serial configuration device is an 8-pin or 16-pin device. The control pins on the serial configuration device are: serial data output (DATA), active serial data input (ASDI), serial clock (DCLK), and chip select (nCS). Table 3–23 lists the serial configuration device's pin descriptions.

Figure 3–21 shows the Altera serial configuration device 8-pin SOIC package and its pin-out diagram.

Figure 3–21. Altera Serial Configuration Device 8-Pin SOIC Package Pin-Out Diagram

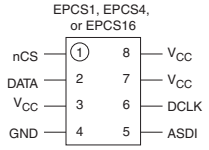
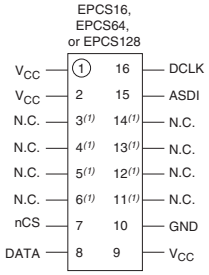


Figure 3–22 shows the Altera serial configuration device 16-pin SOIC package and its pin-out diagram.

Figure 3–22. Altera Serial Configuration Device 16-Pin SOIC Package Pin-Out Diagram



Note to Figure 3–22:

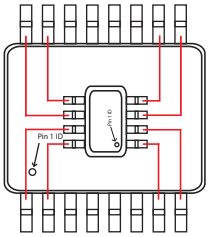
(1) These pins can be left floating or connected to V_{CC} or GND, whichever is more convenient on the board.

Table 3–23. Serial Configuration Device Pin Description

Pin Name	Pin Number in 8-Pin SOIC Package	Pin Number in 16-Pin SOIC Package	Pin Type	Description
DATA	2	8	Output	The DATA output signal transfers data serially out of the serial configuration device to the FPGA during read/configuration operation. During read/configuration operations, the serial configuration device is enabled by pulling *n*CS low. The DATA signal transitions on the falling edge of DCLK.
ASDI	5	15	Input	The AS data input signal is used to transfer data serially into the serial configuration device. It receives the data that should be programmed into the serial configuration device. Data is latched on the rising edge of DCLK.
*n*CS	1	7	Input	The active low chip select input signal toggles at the beginning and end of a valid instruction. When this signal is high, the device is deselected and the DATA pin is tri-stated. When this signal is low, it enables the device and puts the device in an active mode. After power up, the serial configuration device requires a falling edge on the *n*CS signal before beginning any operation.
DCLK	6	16	Input	DCLK is provided by the FPGA. This signal provides the timing of the serial interface. The data presented on ASDI is latched to the serial configuration device on the rising edge of DCLK. Data on the DATA pin changes after the falling edge of DCLK and is latched into the FPGA on the next falling edge.
V_{CC}	3, 7, 8	1, 2, 9	Power	Power pins connect to 3.3 V.
GND	4	10	Ground	Ground pin.

As shown in Figure 3–21 and Figure 3–22, the serial configuration device is an 8-pin or 16-pin device. In order to take advantage of vertical migration from EPSCS1 to EPCS128, Altera recommends a layout for serial configuration devices.

Figure 3–23. Layout Recommendation for Vertical Migration from EPCS1 to EPCS128



Package

EPCS1 and EPCS4 available in 8-pin small outline integrated circuit (SOIC) package. EPCS16 available in 8-pin and 16-pin small outline integrated circuit (SOIC) packages. EPCS64 and EPCS128 available in 16-pin small outline integrated circuit (SOIC) package.

For more information about Altera device packaging including mechanical drawing and specifications for this package, refer to the *Altera Device Package Information Data Sheet*.

Ordering Code

Table 3–24 lists the ordering codes for serial configuration devices.

Table 3–24. Serial Configuration Device Ordering Codes

Device	Ordering Code *(1)*
EPCS1	EPCS1SI8 EPCS1SI8N
EPCS4	EPCS4SI8 EPCS4SI8N
EPCS16	EPCS16SI16N EPCS16SI8N
EPCS64	EPCS64SI16N
EPCS128	EPCS128SI16N

Note to Table 3–24:

(1) N: Lead free.

Document Revision History

Table 3–25 lists the revision history for this chapter.

Table 3–25. Chapter Revision History (Part 1 of 3)

Date	Version	Changes Made
June 2011	3.4	■ Updated Table 3–19. ■ Updated Figure 3–20.
December 2009	3.3	■ Updated "Features" and "Functional Description" sections. ■ Added "Fast Read Operation" section. ■ Removed Table 4–2 to Table 4–9, Table 4-26, and Table 4–33. ■ Updated Table 3–1. ■ Updated Figure 3–2. ■ Removed "Referenced Documents" section.
October 2008	3.2	■ Updated "Introduction", "Active Serial FPGA Configuration", "Operation Codes", "Read Status Operation", "Read Device Identification Operation", and "Package" sections. ■ Updated Table 4–10, Table 4–25, Table 4–26, and Table 4–32. ■ Updated Figure 4–5, Figure 4–13, and Figure 4–19. ■ Added Figure 4–22. ■ Added Table 4–33. ■ Updated new document format.
May 2008	3.1	■ Updated Table 4–3, Table 4–6, Table 4–7, Table 4–28, and Table 4–29. ■ Deleted Note 5 to Table 4–31. ■ Added "Referenced Documents" section.

Table 3–25. Chapter Revision History (Part 2 of 3)

Date	Version	Changes Made
August 2007	3.0	■ Updated “Introduction” section. ■ Updated “Functional Description” section. ■ Updated Table 4–1 through Table 4–4 and Table 4–7 through Table 4–9 to with EPCS128 information. ■ Added Table 4–6 on Arria GX. ■ Added notes to Figure 4–3. ■ Added notes to Figure 4–4. ■ Updated Table 4–10 with EPCS128 information. ■ Added new Table 4–11 on address range for sectors in EPCS128 device. ■ Updated Table 4–16 with information about “Read Device Identification” and added (Note 5). ■ Added new Table 4–21 on block protection bits in EPCS128. ■ Added notes to Figure 4–12. ■ Added new section “Read Device Identification Operation” with Table 4–23 and Figure 4–13. ■ Updated “Write Bytes Operation”, “Erase Bulk Operation” and “Erase Sector Operation” sections. ■ Updated Table 4–24 to include EPCS128 information. ■ Updated (Note 1) to Table 4–26. ■ Updated VCC and VI information to include EPCS128 in Table 4–27. ■ Updated VIH information to include EPCS128 in Table 4–29. ■ Updated ICC0 and ICC1 information to include EPCS128 in Table 4–30. ■ Updated Figure 4–21 and Table 4–34 with EPCS128 information.
April 2007	2.0	■ Updated “Introduction” section. ■ Updated “Functional Description” section and added handpara note. ■ Added Table 4–4, Table 4–6, and Table 4–7. ■ Updated “Active Serial FPGA Configuration” section and its handpara note. ■ Added notes to Figure 4–2. ■ Updated Table 4–26 and added (Note 1). ■ Updated Figure 4–20. ■ Updated Table 4–34.
January 2007	1.7	■ Removed reference to PLMSEPC-16 in “Programming and Configuration File Support”. ■ Updated DCLK pin information in Table 4–32.
October 2006	1.6	■ Updated Figure 4–19. ■ Updated Table 4–30 and Table 4–32.
August 2005	1.5	■ Updated table 4-4 to include EPCS64 support for Cyclone devices.
August 2005	1.4	■ Updated tables. ■ Minor text updates.
February 2005	1.3	Updated hot socketing AC specifications.

Table 3–25. Chapter Revision History (Part 3 of 3)

Date	Version	Changes Made
October 2003	1.2	■ Added Serial Configuration Device Memory Access section. ■ Updated timing information in Tables 4–10 and 4–11 section. ■ Updated timing information in Tables 4-16 and 4-17.
July 2003	1.1	Minor updates.
May 2003	1.0	■ Added document to the *Cyclone Device Handbook.* ■ Initial release.

4. Configuration Devices for SRAM-Based LUT Devices Data Sheet

CF52005-2.4

Features

This chapter describes configuration devices for SRAM-based LUT Devices.

- Configuration device family for configuring Altera® ACEX® 1K, APEX™ 20K (including APEX 20K, APEX 20KC, and APEX 20KE), APEX II, Arria® GX, Cyclone®, Cyclone II, FLEX® 10K (including FLEX 10KE and FLEX 10KA) Mercury™, Stratix®, Stratix GX, Stratix II, and Stratix II GX devices.
- Easy-to-use 4-pin interface to Altera FPGAs
- Low current during configuration and near-zero standby current
- 5.0-V and 3.3-V operation
- Programming support with the Altera Programming Unit (APU) and programming hardware from Data I/O, BP Microsystems, and other third-party programmers
- Available in compact plastic packages
  - 8-pin plastic dual in-line package (PDIP)
  - 20-pin plastic J-lead chip carrier (PLCC) package
  - 32-pin plastic thin quad flat pack (TQFP) package
- EPC2 device has reprogrammable Flash configuration memory
  - 5.0-V and 3.3-V in-system programmability (ISP) through the built-in IEEE Std. 1149.1 JTAG interface
  - Built-in JTAG boundary-scan test (BST) circuitry compliant with IEEE Std. 1149.1
  - Supports programming through Serial Vector Format Files (**.svf**), Jam™ Standard Test and Programming Language (STAPL) Files (**.jam**), Jam STAPL Byte-Code Files (**.jbc**), and the Quartus® II and MAX+PLUS® II softwares via the USB Blaster, MasterBlaster™, ByteBlaster™ II, EthernetBlaster, or ByteBlasterMV™ download cable
  - *n*INIT_CONF pin allows INIT_CONF JTAG instruction to begin FPGA configuration
  - Can be programmed with Programmer Object Files (**.pof**) for EPC1 and EPC1441 devices
  - Available in 20-pin PLCC and 32-pin TQFP packages

For detailed information about enhanced configuration devices, refer to *Enhanced Configuration Devices (EPC4, EPC8 and EPC16) Data Sheet*.

© 2009 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

For detailed information about serial configuration devices, refer to *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64 and EPCS128) Data Sheet*.

Functional Description

With SRAM-based devices, configuration data must be reloaded each time the device powers up, the system initializes, or when new configuration data is needed. Altera configuration devices store configuration data for SRAM-based ACEX 1K, APEX 20K, APEX II, Arria GX, Cyclone, Cyclone II, FLEX 10K, FLEX 6000, Mercury, Stratix, Stratix GX, Stratix II, and Stratix II GX devices. Table 4–1 lists Altera configuration devices and their features.

Table 4–1. Altera Configuration Devices

Device	Memory Size (Bits)	ISP Support	Cascaded Support	Reprogrammable	Operating Voltage
EPC1	1,046,496	No	Yes	No	5.0 or 3.3 V
EPC2	1,695,680	Yes	Yes	Yes	5.0 or 3.3 V
EPC1064	65,536	No	No	No	5.0 V
EPC1064V	65,536	No	No	No	3.3 V
EPC1213	212,942	No	Yes	No	5.0 V
EPC1441	440,800	No	No	No	5.0 or 3.3 V

Table 4–2 lists the supported configuration devices required to configure the ACEX 1K, APEX 20K, APEX 20KC, APEX 20KE, APEX II, Cyclone, Cyclone II, FLEX 10K, FLEX 10KA, FLEX 10KE, FLEX 6000/A , FLEX 8000A , Mercury, Stratix, Stratix GX, or Stratix II device.

Table 4–2. Configuration Devices Required (Part 1 of 4)

Family	Device	Data Size (Bits) *(1)*	EPC1064/ 1064V	EPC1213	EPC1441	EPC1	EPC2
ACEX 1K	EP1K10	159,160	—	—	1	1	1
	EP1K30	473,720	—	—	—	1	1
	EP1K50	784,184	—	—	—	1	1
	EP1K100	1,335,720	—	—	—	—	1
APEX 20K	EP20K100	993,360	—	—	—	1	1
	EP20K200	1,950,800	—	—	—	—	2
	EP20K400	3,880,720	—	—	—	—	3
APEX 20KC	EP20K200C	1,968,016	—	—	—	—	2
	EP20K400C	3,909,776	—	—	—	—	3
	EP20K600C	5,673,936	—	—	—	—	4
	EP20K1000C	8,960,016	—	—	—	—	6

Table 4–2. Configuration Devices Required (Part 2 of 4)

Family	Device	Data Size (Bits) *(1)*	EPC1064/ 1064V	EPC1213	EPC1441	EPC1	EPC2
APEX 20KE	EP20K30E	354,832	—	—	1	1	1
	EP20K60E	648,016	—	—	—	1	1
	EP20K100E	1,008,016	—	—	—	1	1
	EP20K160E	1,524,016	—	—	—	—	1
	EP20K200E	1,968,016	—	—	—	—	2
	EP20K300E	2,741,616	—	—	—	—	2
	EP20K400E	3,909,776	—	—	—	—	3
	EP20K600E	5,673,936	—	—	—	—	4
	EP20K1000E	8,960,016	—	—	—	—	6
	EP20K1500E	12,042,256	—	—	—	—	8
APEX II	EP2A15	1,168,688	—	—	—	—	3
	EP2A25	1,646,544	—	—	—	—	4
	EP2A40	2,543,016	—	—	—	—	6
	EP2A70	4,483,064	—	—	—	—	11
Cyclone	EP1C3	627,376	—	—	—	1	1
	EP1C4	925,000	—	—	—	1	1
	EP1C6	1,167,216	—	—	—	1 *(2)*	1
	EP1C12	2,326,528	—	—	—	—	1 *(2)*
	EP1C20	3,559,608	—	—	—	—	2 *(2)*
Cyclone II	EP2C5	1,265,792	—	—	—	—	1
	EP2C8	1,983,536	—	—	—	—	2
	EP2C20	3,892,496	—	—	—	—	3
	EP2C35	6,848,608	—	—	—	—	5
	EP2C50	9,951,104	—	—	—	—	6
	EP2C70	14,319,216	—	—	—	—	9
FLEX 10K	EPF10K10	118,000	—	—	1	1	1
	EPF10K20	231,000	—	—	1	1	1
	EPF10K30	376,000	—	—	1	1	1
	EPF10K40	498,000	—	—	—	1	1
	EPF10K50	621,000	—	—	—	1	1
	EPF10K70	892,000	—	—	—	1	1
	EPF10K100	1,200,000	—	—	—	—	1
FLEX 10KA	EPF10K10A	120,000	—	—	1	1	1
	EPF10K30A	406,000	—	—	1	1	1
	EPF10K50V	621,000	—	—	—	1	1
	EPF10K100A	1,200,000	—	—	—	—	1
	EPF10K130V	1,600,000	—	—	—	—	1
	EPF10K250A	3,300,000	—	—	—	—	2

Table 4–2. Configuration Devices Required (Part 3 of 4)

Family	Device	Data Size (Bits) *(1)*	EPC1064/ 1064V	EPC1213	EPC1441	EPC1	EPC2
FLEX 10KE	EPF10K30E	473,720	—	—	—	1	1
	EPF10K50E	784,184	—	—	—	1	1
	EPF10K50S	784,184	—	—	—	1	1
	EPF10K100B	1,200,000	—	—	—	—	1
	EPF10K100E	1,335,720	—	—	—	—	1
	EPF10K130E	1,838,360	—	—	—	—	2
	EPF10K200E	2,756,296	—	—	—	—	2
	EPF10K200S	2,756,296	—	—	—	—	2
FLEX 6000/A	EPF6010A	260,000	—	—	1	1	—
	EPF6016 (5.0V) / EPF6016A	260,000	—	—	1	1	—
	EPF6024A	398,000	—	—	1	1	—
FLEX 8000A	EPF8282A / EPF8282AV (3.3 V)	40,000	1	1	1	1	—
	EPF8452A	64,000	1	1	1	1	—
	EPF8636A	96,000	—	1	1	1	—
	EPF8820A	128,000	—	1	1	1	—
	EPF81188A	192,000	—	1	1	1	—
	EPF81500A	250,000	—	—	1	1	—
Mercury	EP1M120	1,303,120	—	—	—	—	1
	EP1M350	4,394,032	—	—	—	—	3
Stratix	EP1S10	3,534,640	—	—	—	—	3 *(3)*
	EP1S20	5,904,832	—	—	—	—	4
	EP1S25	7,894,144	—	—	—	—	5
	EP1S30	10,379,368	—	—	—	—	7
	EP1S40	12,389,632	—	—	—	—	8
	EP1S60	17,543,968	—	—	—	—	11
	EP1S80	23,834,032	—	—	—	—	15
Stratix GX	EP1SGX10	3,534,640	—	—	—	—	3
	EP1SGX25	7,894,144	—	—	—	—	5
	EP1SGX40	12,389,632	—	—	—	—	8

Table 4–2. Configuration Devices Required (Part 4 of 4)

Family	Device	Data Size (Bits) *(1)*	EPC1064/ 1064V	EPC1213	EPC1441	EPC1	EPC2
Stratix II	EP2S15	5,000,000	—	—	—	—	3
	EP2S30	10,100,000	—	—	—	—	7
	EP2S60	17,100,000	—	—	—	—	11
	EP2S90	27,500,000	—	—	—	—	17
	EP2S130	39,600,000	—	—	—	—	24
	EP2S180	52,400,000	—	—	—	—	31

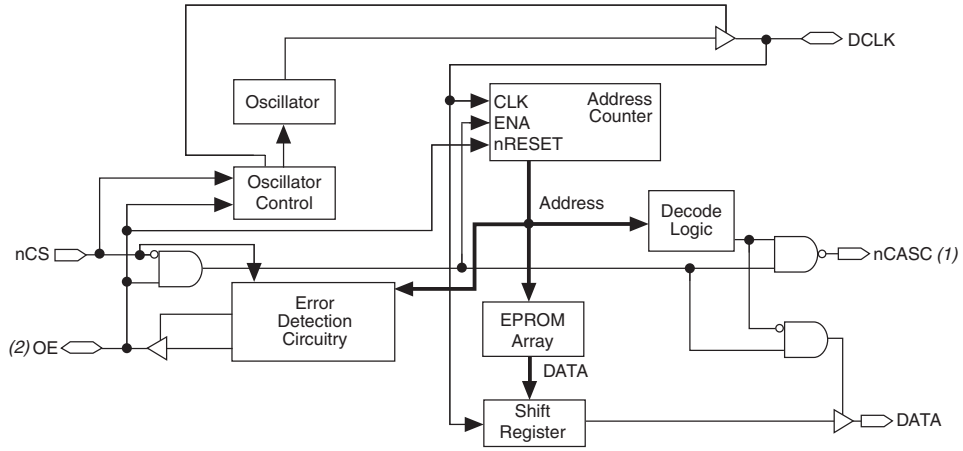
Notes to Table 4–2:

(1) Raw Binary Files (**.rbf**) were used to determine these sizes.
(2) This is with the Cyclone series compression feature enabled.
(3) EP1S10 ES devices requires four EPC2 devices.

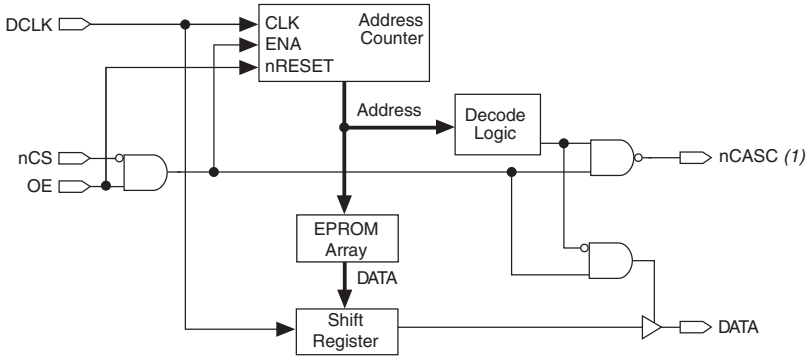
Figure 4–1 shows the configuration device block diagram.

Figure 4–1. Configuration Device Block Diagram

FPGA (except FLEX 8000) Configuration Using an EPC2, EPC1, or EPC1441



FLEX 8000 Device Configuration Using an EPC1, EPC1441, EPC1213, EPC1064, or EPC1064V



Notes to Figure 4–1:

(1) The EPC1441 devices do not support data cascading. The EPC1, EPC2, and EPC1213 devices support data cascading.
(2) The OE pin is a bidirectional open-drain pin.

Device Configuration

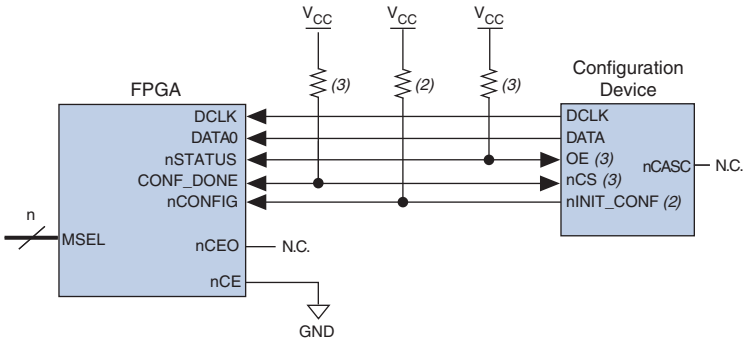
The EPC1, EPC2, and EPC1441 devices store configuration data in its EPROM array and serially clock data out using an internal oscillator. The OE, nCS, and DCLK pins supply the control signals for the address counter and the DATA output tri-state buffer. The configuration device sends a serial bitstream of configuration data to its DATA pin, which is routed to the DATA0 input of the FPGA.

The control signals for configuration devices (OE, nCS, and DCLK) interface directly with the FPGA control signals (nSTATUS, CONF_DONE, and DCLK, respectively). All Altera FPGAs can be configured by a configuration device without requiring an external intelligent controller.

An EPC2 device cannot configure FLEX 8000 or FLEX 6000 devices. Refer to Table 4–2 on page 4–2 for the configuration devices that support FLEX 8000 and FLEX 6000 devices.

Figure 4–2 shows the basic configuration interface connections between the configuration device and the Altera FPGA. For specific details about configuration interface connections, including pull-up resistor values, supply voltages and MSEL pin setting, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.

Figure 4–2. Altera FPGA Configured Using an EPC1, EPC2, or EPC1441 Configuration Device *(Note 1)*



Notes to Figure 4–2:

(1) For specific details about configuration interface connections refer to the FPGA family chapter in the *Configuration Handbook*.

(2) The nINIT_CONF pin (available on EPC2 devices) has an internal pull-up resistor that is always active. This means an external pull-up resistor is not required on the nINIT_CONF/nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available, nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) EPC2 devices have internal programmable pull-up resistors on OE and nCS. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

The EPC2 device allows the user to begin configuration of the FPGA via an additional pin, nINIT_CONF. The nINIT_CONF pin of the EPC2 device can be connected to the nCONFIG of the FPGA, which allows the INIT_CONF JTAG instruction to begin FPGA configuration. The INIT_CONF JTAG instruction causes the EPC2 device to drive nINIT_CONF low, which in turn pulls nCONFIG low. Pulling nCONFIG low on the FPGA will reset the device. When the JTAG state machine exits this state, nINIT_CONF is released and pulled high by an internal 1-kΩ resistor, which in turn pulls nCONFIG high to begin configuration. If its functionality is not used, the nINIT_CONF pin does not need to be connected and nCONFIG of the FPGA must be pulled to V_{CC} either directly or through a resistor.

The EPC2 device's OE and *n*CS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS** and **OE** pull-ups on configuration device option when generating programming files.

The configuration device's OE and *n*CS pins control the tri-state buffer on its DATA output pin, and enable the address counter and oscillator. When OE is driven low, the configuration device resets the address counter and tri-states its DATA pin. The *n*CS pin controls the DATA output of the configuration device. If *n*CS is held high after the OE reset pulse, the counter is disabled and the DATA output pin is tri-stated. If *n*CS is driven low after the OE reset pulse, the counter and DATA output pin are enabled. When OE is driven low again, the address counter is reset and the DATA output pin is tri-stated, regardless of the state of *n*CS.

If the FPGA's configuration data exceeds the capacity of a single EPC1 or EPC2 configuration device, multiple EPC1 or EPC2 devices can be cascaded together. If multiple EPC1 or EPC2 devices are required, the *n*CASC and *n*CS pins provide handshaking between the configuration devices.

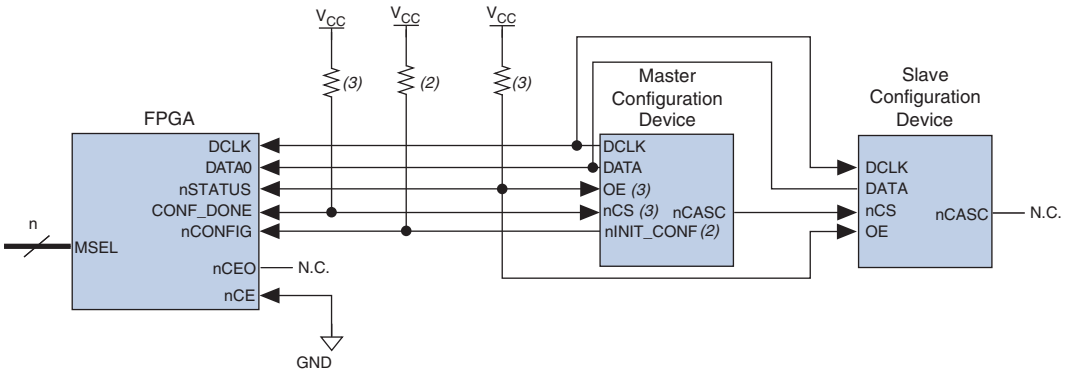
EPC1441 and EPC1064/V devices cannot be cascaded.

When configuring ACEX 1K, APEX 20K, APEX II, Arria GX, Cyclone, Cyclone II, FLEX 10K, Mercury, Stratix, Stratix GX, Stratix II, and Stratix II GX devices with cascaded EPC1 or EPC2 devices, the position of the EPC1 or EPC2 device in the chain determines its mode of operation. The first configuration device in the chain is the master, while subsequent configuration devices are slaves. The *n*INIT_CONF pin of the master EPC2 device can be connected to the *n*CONFIG of the FPGAs, which allows the INIT_CONF JTAG instruction to begin FPGA configuration. The *n*CS pin of the master configuration device is connected to the CONF_DONE of the FPGAs, while its *n*CASC pin is connected to *n*CS of the next slave configuration device in the chain. Additional EPC1 or EPC2 devices can be chained together by connecting *n*CASC to *n*CS of the next slave EPC1 or EPC2 device in the chain. The last device's *n*CS input comes from the previous device, while its *n*CASC pin is left floating. All other configuration pins (DCLK, DATA, and OE) are connected to every device in the chain.

Figure 4–3 shows the basic configuration interface connections between a configuration device chain and the Altera FPGA.

For specific details about configuration interface connections, including pull-up resistor values, supply voltages and MSEL pin setting, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.

Figure 4–3. Altera FPGA Configured Using Two EPC1 or EPC2 Configuration Devices ***(Note 1)***



Notes to Figure 4–3:

(1) For specific details about configuration interface connections refer to the appropriate FPGA family chapter in the *Configuration Handbook*.

(2) The nINIT_CONF pin (available on EPC2 devices) has an internal pull-up resistor that is always active. This means an external pull-up resistor is not required on the nINIT_CONF/nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used or not available, nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(3) EPC2 devices have internal programmable pull-up resistors on OE and nCS. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

When the first device in a configuration device chain is powered-up or reset, its nCS pin is driven low because it is connected to the CONF_DONE of the FPGA. Because both OE and nCS are low, the first device in the chain will recognize it is the master device and will control configuration. Since the slave devices' nCS pin is fed by the previous devices' nCASC pin, its nCS pin will be high upon power-up and reset. In the slave configuration devices, the DATA output is tri-stated and DCLK is an input. During configuration, the master device supplies the clock through DCLK to the FPGA and to any slave configuration devices. The master EPC1 or EPC2 device also provides the first stream of data to the FPGA during multi-device configuration. After the master EPC1 or EPC2 device finishes sending configuration data, it tri-states its DATA pin to avoid contention with other configuration devices. The master EPC1 or EPC2 device will also drive its nCASC pin low, which pulls the nCS pin of the next device low. This action signals the slave EPC1 or EPC2 device to start sending configuration data to the FPGAs.

The master EPC1 or EPC2 device clocks all slave configuration devices until configuration is complete. When all configuration data is transferred and the nCS pin on the master EPC1 or EPC2 device is driven high by the FPGA's CONF_DONE pin, the master EPC1 or EPC2 device then goes into zero-power (idle) state. The master EPC2 device drives DATA high and DCLK low, while the EPC1 and EPC1441 device tri-state DATA and drive DCLK low.

If nCS on the master EPC1 or EPC2 device is driven high before all configuration data is transferred, the master EPC1 or EPC2 device drives its OE signal low, which in turn drives the FPGA's nSTATUS pin low, indicating a configuration error. Additionally, if the configuration device sends is generated its data and detects that CONF_DONE has not gone high, it recognizes that the FPGA has not configured successfully. EPC1 and EPC2 devices wait for 16 DCLK cycles after the last configuration bit was sent for

CONF_DONE to reach a high state. In this case, the configuration device pulls its OE pin low, which in turn drives the target device's nSTATUS pin low. Configuration automatically restarts if the **Auto-restart configuration on error** option is turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box or the MAX+PLUS II software's **Global Project Device Options** dialog box (Assign menu).

For more information about FPGA configuration and configuration interface connections between configuration devices and Altera FPGAs, refer to the appropriate FPGA family chapter in the *Configuration Handbook*.

Power and Operation

This section describes Power-On Reset (POR) delay, error detection, and 3.3-V and 5.0-V operation of Altera configuration devices.

Power-On Reset (POR)

During initial power-up, a POR delay occurs to permit voltage levels to stabilize. When configuring an FPGA with an EPC1, EPC2, or EPC1441 device, the POR delay occurs inside the configuration device, and the POR delay is a maximum of 200 ms. When configuring a FLEX 8000 device with an EPC1213, EPC1064, or EPC1064V device, the POR delay occurs inside the FLEX 8000 device, and the POR delay is typically, 100 ms, with a maximum of 200 ms.

During POR, the configuration device drives its OE pin low. This low signal delays configuration because the OE pin is connected to the target FPGA's nSTATUS pin. When the configuration device completes POR, it releases its open-drain OE pin, which is then pulled high by a pull-up resistor.

The FPGA should be powered up before the configuration device exits POR to avoid the master configuration device from entering slave mode.

If the FPGA is not powered up before the configuration device exits POR, the CONF_DONE/nCS line will be high because of the pull-up resistor. When the configuration device exits POR and releases OE, it sees nCS high, which signals the configuration device to enter slave mode. Therefore, configuration will not begin (the DATA output is tri-stated and DCLK is an input pin in slave mode).

Error Detection Circuitry

The EPC1, EPC2, and EPC1441 configuration devices have built-in error detection circuitry for configuring ACEX 1K, APEX 20K, APEX II, Arria GX, Cyclone, Cyclone II, FLEX 10K, FLEX 6000, Mercury, Stratix, Stratix GX, Stratix II, or Stratix II GX devices.

Built-in error-detection circuitry uses the nCS pin of the configuration device, which monitors the CONF_DONE pin on the FPGA. If nCS on the master EPC1 or EPC2 device is driven high before all configuration data is transferred, the master EPC1 or EPC2 device drives its OE signal low, which in turn drives the FPGA's nSTATUS pin low, indicating a configuration error. Additionally, if the configuration device sends is generated its data and detects that CONF_DONE has not gone high, it recognizes that the

FPGA has not configured successfully. EPC1 and EPC2 devices wait for 16 DCLK cycles after the last configuration bit was sent for CONF_DONE to reach a high state. In this case, the configuration device pulls its OE pin low, which in turn drives the target device's nSTATUS pin low. Configuration automatically restarts if the **Auto-restart configuration on error** option is turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box or the MAX+PLUS II software's **Global Project Device Options** dialog box (Assign menu).

In addition, if the FPGA detects a cyclical redundancy code (CRC) error in the received data, it will flag the error by driving nSTATUS low. This low signal on nSTATUS will drive the OE pin of the configuration device low, which will reset the configuration device. CRC checking is performed when configuring all Altera FPGAs.

3.3-V or 5.0-V Operation

The EPC1, EPC2, and EPC 1441 configuration device may be powered at 3.3 V or 5.0 V. For each configuration device, an option must be set for 5.0-V or 3.3-V operation.

For EPC1 and EPC1441 configuration devices, 3.3-V or 5.0-V operation is controlled by a programming bit in the **.pof**. The **Low-Voltage mode** option in the **Options** tab of the **Configuration Device Options** dialog box in the Quartus II software or the **Use Low-Voltage Configuration EPROM** option in the **Global Project Device Options** dialog box (Assign menu) in the MAX+PLUS II software sets this parameter. For example, EPC1 devices are programmed automatically to operate in 3.3-V mode when configuring FLEX 10KA devices, which have a V_{CC} voltage of 3.3 V. In this example, the EPC1 device's VCC pin is connected to a 3.3-V power supply.

For EPC2 devices, this option is set externally by the VCCSEL pin. In addition, the EPC2 device has an externally controlled option, set by the VPPSEL pin, to adjust the programming voltage to 5.0 V or 3.3 V. The functions of the VCCSEL and VPPSEL pins are described below. These pins are only available in the EPC2 devices.

- VCCSEL pin – For EPC2 configuration devices, 5.0-V or 3.3-V operation is controlled by the VCCSEL option pin. The device functions in 5.0-V mode when VCCSEL is connected to GND; the device functions in 3.3-V mode when VCCSEL is connected to V_{CC}.
- VPPSEL pin – The EPC2 V_{PP} programming power pin is normally tied to V_{CC}. For EPC2 devices operating at 3.3 V, it is possible to improve in-system programming times by setting V_{PP} to 5.0 V. For all other configuration devices, V_{PP} must be tied to V_{CC}. The EPC2 device's VPPSEL pin must be set in accordance with the EPC2 VPP pin. If the VPP pin is supplied by a 5.0-V supply, VPPSEL must be connected to GND; if the VPP pin is supplied by a 3.3-V power supply, VPPSEL must be connected to V_{CC}.

Table 4–3 describes the relationship between the V_{CC} and V_{PP} voltage levels and the required logic level for VCCSEL and VPPSEL. A logic level of high means the pin should be connected to V_{CC}, while a low logic level means the pin should be connected to GND.

Table 4–3. VCCSEL and VPPSEL Pin Functions on the EPC2

VCC Voltage Level (V)	VPP Voltage Level (V)	VCCSEL Pin Logic Level	VPPSEL Pin Logic Level
3.3	3.3	High	High
3.3	5.0	High	Low
5.0	5.0	Low	Low

At 3.3-V operation, all EPC2 inputs are 5.0-V tolerant, except DATA, DCLK, and *n*CASC. The DATA and DCLK pins are used only to interface between the EPC2 device and the FPGA it is configuring. The voltage tolerances of all EPC2 pins at 5.0 V and 3.3 V are listed in Table 4–4.

Table 4–4. EPC2 Input and Bidirectional Pin Voltage Tolerance

Pin	5.0-V Operation		3.3-V Operation	
	5.0-V Tolerant	3.3-V Tolerant	5.0-V Tolerant	3.3-V Tolerant
DATA	✓	✓	—	✓
DCLK	✓	✓	—	✓
*n*CASC	✓	✓	—	✓
OE	✓	✓	✓	✓
*n*CS	✓	✓	✓	✓
VCCSEL	✓	✓	✓	✓
VPPSEL	✓	✓	✓	✓
*n*INIT_CONF	✓	✓	✓	✓
TDI	✓	✓	✓	✓
TMS	✓	✓	✓	✓
TCK	✓	✓	✓	✓

If an EPC1, EPC2, or EPC1441 configuration device is powered at 3.3 V, the *n*STATUS and CONF_DONE pull-up resistors must be connected to 3.3 V. If these configuration devices are powered at 5.0 V, the *n*STATUS and CONF_DONE pull-up resistors can be connected to 3.3 V or 5.0 V.

Programming and Configuration File Support

The Quartus II and MAX+PLUS II softwares provide programming support for Altera configuration devices. During compilation, the Quartus II and MAX+PLUS II softwares automatically generates a **.pof**, which can be used to program the configuration devices. In a multi-device project, the software can combine the programming files for multiple ACEX 1K, APEX 20K, APEX II, Arria GX, Cyclone, Cyclone II, FLEX 10K, Mercury, Stratix, Stratix GX, Stratix II, and Stratix II GX devices into one or more configuration devices. The software allows you to select the appropriate configuration device to most efficiently store the data for each FPGA.

All Altera configuration devices are programmable using Altera programming hardware in conjunction with the Quartus II or MAX+PLUS II software. In addition, many third part programmers offer programming hardware that supports Altera configuration devices.

An EPC2 device can be programmed with a **.pof** generated for an EPC1 or EPC1441 device. An EPC1 device can be programmed using a **.pof** generated for an EPC1441 device.

EPC2 configuration devices can be programmed in-system through its industry-standard 4-pin JTAG interface. ISP capability in the EPC2 devices provides ease in prototyping and FPGA functionality. When programming multiple EPC2 devices in a JTAG chain, the Quartus II and MAX+PLUS II softwares and other programming methods employ concurrent programming to simultaneously program multiple devices and reduce programming time. EPC2 devices can be programmed and erased up to 100 times.

After programming an EPC2 device in-system, FPGA configuration can be initiated by the EPC2 `INIT_CONF` JTAG instruction. Refer to Table 4–6 on page 4–13.

For more information about programming and configuration support, refer to the following documents:

- *Altera Programming Hardware Data Sheet*
- *USB-Blaster Download Cable User Guide*
- *MasterBlaster Serial/USB Communications Cable User Guide*
- *ByteBlaster II Download Cable User Guide*
- *ByteBlasterMV Download Cable User Guide*
- *BitBlaster Serial Download Cable Data Sheet*

You can also program configuration devices using the Quartus II or MAX+PLUS II software with the APU, and the appropriate configuration device programming adapter.

Table 4–5 shows which programming adapter to use with each configuration device.

Table 4–5. Programming Adapters

Device	Package	Adapter
EPC2	20-pin J-Lead	PLMJ1213
	32-pin TQFP	PLMT1213
EPC1	8-pin DIP	PLMJ1213
	20-pin J-Lead	PLMJ1213
EPC1441	8-pin DIP	PLMJ1213
	20-pin J-Lead	PLMJ1213
	32-pin TQFP	PLMT1064

The following steps explain how to program Altera configuration devices using the Quartus II software and the APU:

1. Choose the **Quartus II Programmer** (Tools menu).
2. Load the appropriate **.pof** by clicking **Add**. The **Device** column displays the device for the current programming file.
3. Insert a blank configuration device into the programming adapter's socket.
4. Turn on the **Program/Configure**. You can also turn on **Verify** to verify the contents of a programmed device against the programming data loaded from a programming file.
5. Click **Start**.
6. After successful programming, you can place the configuration device on the PCB to configure the FPGA device.

The following steps explain how to program Altera configuration devices using the MAX+PLUS II software and the APU:

1. Open the MAX+PLUS II Programmer.
2. Load the appropriate **.pof** using the **Select Programming File** dialog box (File menu). By default, the **Programmer** loads the current project's **.pof**. The **Device** field displays the device for the current programming file.
3. Insert a blank configuration device into the programming adapter's socket.
4. Click **Program**.
5. After successful programming, you can place the configuration device on the PCB to configure the FPGA device.

If you are cascading EPC1 or EPC2 devices, you must generate multiple **.pof**. The first device **.pof** will have the same name as the project, while the second device **.pof** will have the same name as the first, but with a "_1" extension (e.g., **top_1.pof**).

IEEE Std. 1149.1 (JTAG) Boundary-Scan Testing

The EPC2 provides JTAG BST circuitry that complies with the IEEE Std. 1149.1-1990 specification. JTAG boundary-scan testing can be performed before or after configuration, but not during configuration. The EPC2 device supports the JTAG instructions shown in Table 4–6.

Table 4–6. EPC2 JTAG Instructions (Part 1 of 2)

JTAG Instruction	OPCODE	Description
SAMPLE/PRELOAD	00 0101 0101	Allows a snapshot of a signal at the device pins to be captured and examined during normal device operation, and permits an initial data pattern output at the device pins.
EXTEST	00 0000 0000	Allows the external circuitry and board-level interconnections to be tested by forcing a test pattern at the output pins and capturing results at the input pins.
BYPASS	11 1111 1111	Places the 1-bit bypass register between the TDI and TDO pins, which allows the BST data to pass synchronously through a selected device to adjacent devices during normal device operation.

Table 4–6. EPC2 JTAG Instructions (Part 2 of 2)

JTAG Instruction	OPCODE	Description
IDCODE	00 0101 1001	Selects the device IDCODE register and places it between TDI and TDO, allowing the device IDCODE to be serially shifted out of TDO. The device IDCODE for the EPC2 configuration device is shown below: 0000 0001000000000010 00001101110 1
USERCODE	00 0111 1001	Selects the USERCODE register and places it between TDI and TDO, allowing the USERCODE to be serially shifted out of TDO. The 32-bit USERCODE is a programmable user-defined pattern.
INIT_CONF	00 0110 0001	This function initiates the FPGA re-configuration process by pulsing the *n*INIT_CONF pin low, which is connected to the FPGAs *n*CONFIG pins. After this instruction is updated, the *n*INIT_CONF pin is pulsed low when the JTAG state machine enters the Run-Test/Idle state. The *n*INIT_CONF pin is then released and nCONFIG is pulled high by the resistor after the JTAG state machine goes out of Run-Test/Idle state. The FPGA configuration starts after *n*CONFIG goes high. As a result, the FPGA is configured with the new configuration data stored in the configuration device. This function can be added to your programming file (**.pof**, **.jam**, **.jbc**) in the Quartus II software by enabling the **Initiate configuration after programming** option in the **Programmer options** window (Options menu). This instruction is also used by the MAX+PLUS II software, **.jam** files, and **.jbc** files.
ISP Instructions	—	These instructions are used when programming an EPC2 device via JTAG ports with a USB Blaster, MasterBlaster, ByteBlaster II, EthernetBlaster, or ByteBlaster MV download cable, or using a **.jam**, **.jbc**, or **.svf** via an embedded processor.

For more information, refer to *Application Note 39 (IEEE 1149.1 (JTAG) Boundary-Scan Testing in Altera Devices)* or the EPC2 BSDL files on the Altera web site.

Figure 4–4 shows the timing requirements for the JTAG signals.

Figure 4–4. EPC2 JTAG Waveforms

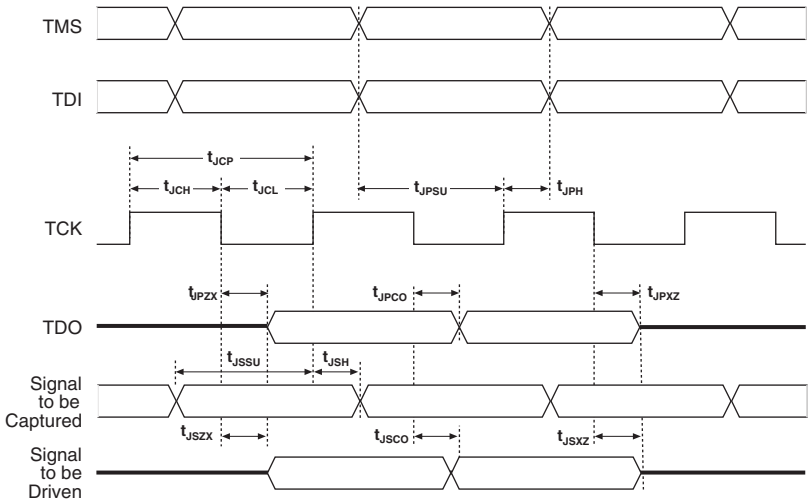


Table 4–7 shows the timing parameters and values for configuration devices.

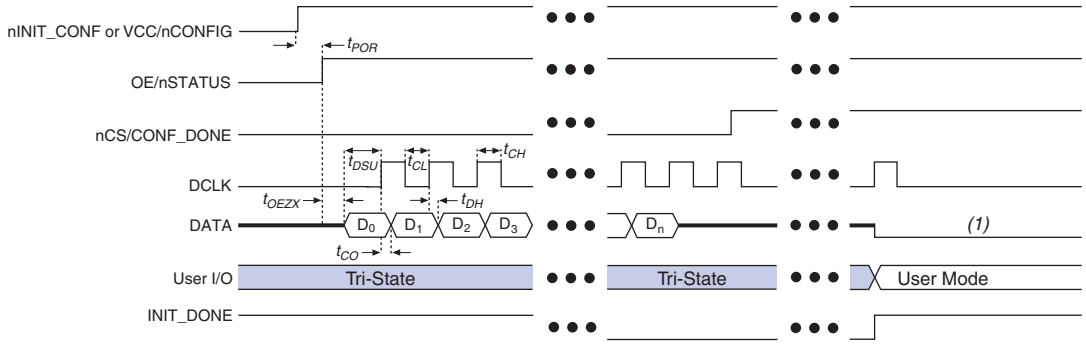
Table 4–7. JTAG Timing Parameters and Values

Symbol	Parameter	Min	Max	Unit
t_{JCP}	TCK clock period	100	—	ns
t_{JCH}	TCK clock high time	50	—	ns
t_{JCL}	TCK clock low time	50	—	ns
t_{JPSU}	JTAG port setup time	20	—	ns
t_{JPH}	JTAG port hold time	45	—	ns
t_{JPCO}	JTAG port clock to output	—	25	ns
t_{JPZX}	JTAG port high impedance to valid output	—	25	ns
t_{JPXZ}	JTAG port valid output to high impedance	—	25	ns
t_{JSSU}	Capture register setup time	20	—	ns
t_{JSH}	Capture register hold time	45	—	ns
t_{JSCO}	Update register clock to output	—	25	ns
t_{JSZX}	Update register high-impedance to valid output	—	25	ns
t_{JSXZ}	Update register valid output to high impedance	—	25	ns

Timing Information

Figure 4–5 shows the timing waveform when using a configuration device.

Figure 4–5. Timing Waveform Using a Configuration Device



Note to Figure 4–5:

(1) The EPC2 device will drive DCLK low and DATA high after configuration. The EPC1 and EPC1441 device will drive DCLK low and tri-state DATA after configuration.

Table 4–8 defines the timing parameters when using EPC2 devices at 3.3 V.

Table 4–8. Timing Parameters when Using EPC2 devices at 3.3 V

Symbol	Parameter	Min	Typ	Max	Units
t_{POR}	POR delay *(1)*	—	—	200	ms
t_{OEZX}	OE high to DATA output enabled	—	—	80	ns
t_{CE}	OE high to first rising edge on DCLK	—	—	300	ns
t_{DSU}	Data setup time before rising edge on DCLK	30	—	—	ns
t_{DH}	Data hold time after rising edge on DCLK	0	—	—	ns
t_{CO}	DCLK to DATA out	—	—	30	ns
t_{CDOE}	DCLK to DATA enable/disable	—	—	30	ns
f_{CLK}	DCLK frequency	5	7.7	12.5	MHz
t_{MCH}	DCLK high time for the first device in the configuration chain	40	65	100	ns
t_{MCL}	DCLK low time for the first device in the configuration chain	40	65	100	ns
t_{SCH}	DCLK high time for subsequent devices	40	—	—	ns
t_{SCL}	DCLK low time for subsequent devices	40	—	—	ns
t_{CASC}	DCLK rising edge to *n*CASC	—	—	25	ns
t_{CCA}	*n*CS to nCASC cascade delay	—	—	15	ns
t_{OEW}	OE low pulse width (reset) to guarantee counter reset	100	—	—	ns
t_{OEC}	OE low (reset) to DCLK disable delay	—	—	30	ns
t_{NRCAS}	OE low (reset) to *n*CASC delay	—	—	30	ns

Note to Table 4–8:

(1) During initial power-up, a POR delay occurs to permit voltage levels to stabilize. Subsequent reconfigurations do not incur this delay.

Table 4–9 defines the timing parameters when using EPC1 and EPC1441 devices at 3.3 V.

Table 4–9. Timing Parameters when Using EPC1 and EPC1441 Devices at 3.3 V (Part 1 of 2)

Symbol	Parameter	Min	Typ	Max	Units
t_{POR}	POR delay *(1)*	—	—	200	ms
t_{OEZX}	OE high to DATA output enabled	—	—	80	ns
t_{CE}	OE high to first rising edge on DCLK	—	—	300	ns
t_{DSU}	Data setup time before rising edge on DCLK	30	—	—	ns
t_{DH}	Data hold time after rising edge on DCLK	0	—	—	ns
t_{CO}	DCLK to DATA out	—	—	30	ns
t_{CDOE}	DCLK to DATA enable/disable	—	—	30	ns
f_{CLK}	DCLK frequency	2	4	10	MHz
t_{MCH}	DCLK high time for the first device in the configuration chain	50	125	250	ns
t_{MCL}	DCLK low time for the first device in the configuration chain	50	125	250	ns
t_{SCH}	DCLK high time for subsequent devices	50	—	—	ns
t_{SCL}	DCLK low time for subsequent devices	50	—	—	ns

Table 4–9. Timing Parameters when Using EPC1 and EPC1441 Devices at 3.3 V (Part 2 of 2)

Symbol	Parameter	Min	Typ	Max	Units
t_{CASC}	DCLK rising edge to *n*CASC	—	—	25	ns
t_{CCA}	*n*CS to *n*CASC cascade delay	—	—	15	ns
t_{OEW}	OE low pulse width (reset) to guarantee counter reset	100	—	—	ns
t_{OEC}	OE low (reset) to DCLK disable delay	—	—	30	ns
t_{NRCAS}	OE low (reset) to *n*CASC delay	—	—	30	ns

Note to Table 4–9:

(1) During initial power-up, a POR delay occurs to permit voltage levels to stabilize. Subsequent reconfigurations do not incur this delay.

Table 4–10 defines the timing parameters when using EPC1, EPC2, and EPC1441 devices at 5.0 V.

Table 4–10. Timing Parameters when Using EPC1, EPC2 and EPC1441 Devices at 5.0 V

Symbol	Parameter	Min	Typ	Max	Units
t_{POR}	POR delay *(1)*	—	—	200	ms
t_{OEZX}	OE high to DATA output enabled	—	—	50	ns
t_{CE}	OE high to first rising edge on DCLK	—	—	200	ns
t_{DSU}	Data setup time before rising edge on DCLK	30	—	—	ns
t_{DH}	Data hold time after rising edge on DCLK	0	—	—	ns
t_{CO}	DCLK to DATA out	—	—	20	ns
t_{CDOE}	DCLK to DATA enable/disable	—	—	20	ns
f_{CLK}	DCLK frequency	6.7	10	16.7	MHz
t_{MCH}	DCLK high time for the first device in the configuration chain	30	50	75	ns
t_{MCL}	DCLK low time for the first device in the configuration chain	30	50	75	ns
t_{SCH}	DCLK high time for subsequent devices	30	—	—	ns
t_{SCL}	DCLK low time for subsequent devices	30	—	—	ns
t_{CASC}	DCLK rising edge to *n*CASC	—	—	20	ns
t_{CCA}	nCS to *n*CASC cascade delay	—	—	10	ns
t_{OEW}	OE low pulse width (reset) to guarantee counter reset	100	—	—	ns
t_{OEC}	OE low (reset) to DCLK disable delay	—	—	20	ns
t_{NRCAS}	OE low (reset) to *n*CASC delay	—	—	25	ns

Note to Table 4–10:

(1) During initial power-up, a POR delay occurs to permit voltage levels to stabilize. Subsequent reconfigurations do not incur this delay.

Table 4–11 defines the timing parameters when using EPC1, EPC1064, EPC1064V, EPC1213, and EPC1441 devices when configuring FLEX 8000 device.

Table 4–11. FLEX 8000 Device Configuration Parameters Using EPC1, EPC1064, EPC1064V, EPC1213, and EPC1441 Devices

Symbol	Parameter	EPC1064V		EPC1064 EPC1213		EPC1 EPC1441		Unit
		Min	Max	Min	Max	Min	Max	
t_{OEZX}	OE high to DATA output enabled	—	75	—	50	—	50	ns
t_{CSZX}	*n*CS low to DATA output enabled	—	75	—	50	—	50	ns
t_{CSXZ}	*n*CS high to DATA output disabled	—	75	—	50	—	50	ns
t_{CSS}	*n*CS low setup time to first DCLK rising edge	150	—	100	—	50	—	ns
t_{CSH}	*n*CS low hold time after DCLK rising edge	0	—	0	—	0	—	ns
t_{DSU}	Data setup time before rising edge on DCLK	75	—	50	—	50	—	ns
t_{DH}	Data hold time after rising edge on DCLK	0	—	0	—	0	—	ns
t_{CO}	DCLK to DATA out delay	—	100	—	75	—	75	ns
t_{CK}	Clock period	240	—	160	—	100	—	ns
f_{CK}	Clock frequency	—	4	—	6	—	8	MHz
t_{CL}	DCLK low time	120	—	80	—	50	—	ns
t_{CH}	DCLK high time	120	—	80	—	50	—	ns
t_{XZ}	OE low or *n*CS high to DATA output disabled	—	75	—	50	—	50	ns
t_{OEW}	OE pulse width to guarantee counter reset	150	—	100	—	100	—	ns
t_{CASC}	Last DCLK + 1 to *n*CASC low delay	—	90	—	60	—	50	ns
t_{CKXZ}	Last DCLK + 1 to DATA tri-state delay	—	75	—	50	—	50	ns
t_{CEOUT}	*n*CS high to *n*CASC high delay	—	150	—	100	—	100	ns

Operating Conditions

Table 4–12 through Table 4–19 provide information about absolute maximum ratings, recommended operating conditions, DC operating conditions, and capacitance for configuration devices.

Table 4–12. Absolute Maximum Ratings ***(Note 1)***

Symbol	Parameter	Conditions	Min	Max	Unit
V_{CC}	Supply voltage	With respect to ground *(2)*	–2.0	7.0	V
V_I	DC input voltage	With respect to ground *(2)*	–2.0	7.0	V
I_{MAX}	DC V_{CC} or ground current	—	—	50	mA
I_{OUT}	DC output current, per pin	—	–25	25	mA
P_D	Power dissipation	—	—	250	mW
T_{STG}	Storage temperature	No bias	–65	150	° C
T_{AMB}	Ambient temperature	Under bias	–65	135	° C
T_J	Junction temperature	Under bias	—	135	° C

Table 4–13. Recommended Operating Conditions

Symbol	Parameter	Conditions	Min	Max	Unit
V_{CC}	Supply voltage for 5.0-V operation	*(3), (4)*	4.75 (4.50)	5.25 (5.50)	V
	Supply voltage for 3.3-V operation	*(3), (4)*	3.0 (3.0)	3.6 (3.6)	V
V_I	Input voltage	With respect to ground	–0.3	V_{CC} + 0.3 *(5)*	V
V_O	Output voltage	—	0	V_{CC}	V
T_A	Operating temperature	For commercial use	0	70	° C
		For industrial use	–40	85	° C
t_R	Input rise time	—	—	20	ns
t_F	Input fall time	—	—	20	ns

Table 4–14. DC Operating Conditions

Symbol	Parameter	Conditions	Min	Max	Unit
V_{IH}	High-level input voltage	—	2.0	V_{CC} + 0.3 *(5)*	V
V_{IL}	Low-level input voltage	—	–0.3	0.8	V
V_{OH}	5.0-V mode high-level TTL output voltage	I_{OH} = –4 mA DC *(6)*	2.4	—	V
	3.3-V mode high-level CMOS output voltage	I_{OH} = –0.1 mA DC *(6)*	V_{CC} – 0.2	—	V
V_{OL}	Low-level output voltage	I_{OL} = 4 mA DC *(6)*	—	0.4	V
I_I	Input leakage current	$V_I = V_{CC}$ or ground	–10	10	µA
I_{OZ}	Tri-state output off-state current	$V_O = V_{CC}$ or ground	–10	10	µA

Table 4–15. EPC1064, EPC1064V, and EPC1213 Device I_{CC} Supply Current Values

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
I_{CC0}	V_{CC} supply current (standby)	—	—	100	200	µA
I_{CC1}	V_{CC} supply current (during configuration)	—	—	10	50	mA

Table 4–16. EPC2 Device Values

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
I_{CC0}	V_{CC} supply current (standby)	V_{CC} = 5.0 V or 3.3 V	—	50	100	µA
I_{CC1}	V_{CC} supply current (during configuration)	V_{CC} = 5.0 V or 3.3 V	—	18	50	mA
R_{CONF}	Configuration pins	Internal pull up (OE, *n*CS, *n*INIT_CONF)	—	1	—	kΩ

Table 4–17. EPC1 Device I_{CC} Supply Current Values

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
I_{CC0}	V_{CC} supply current (standby)	—	—	50	100	µA
I_{CC1}	V_{CC} supply current (during configuration)	V_{CC} = 5.0 V	—	30	50	mA
		V_{CC} = 3.3 V	—	10	16.5	mA

Table 4–18. EPC1441 Device I_{CC} Supply Current Values

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
I_{CC0}	V_{CC} supply current (standby)	—	—	30	60	µA
I_{CC1}	V_{CC} supply current (during configuration)	V_{CC} = 5.0 V	—	15	30	mA
I_{CC1}	V_{CC} supply current (during configuration)	V_{CC} = 3.3 V	—	5	10	mA

Table 4–19. Capacitance *(Note 7)*

Symbol	Parameter	Conditions	Min	Max	Unit
C_{IN}	Input pin capacitance	V_{IN} = 0 V, f = 1.0 MHz	—	10	pF
C_{OUT}	Output pin capacitance	V_{OUT} = 0 V, f = 1.0 MHz	—	10	pF

Notes to Table 4–12 through Table 4–19:

(1) For more information, refer to the Operating Requirements for Altera Devices Data Sheet.

(2) The minimum DC input is -0.3 V. During transitions, the inputs may undershoot to -2.0 V or overshoot to 7.0 V for input currents less than 100 mA and periods shorter than 20 ns under no-load conditions.

(3) Numbers in parentheses are for industrial temperature range devices.

(4) Maximum V_{CC} rise time is 100 ms.

(5) Certain EPC2 pins may be driven to 5.75 V when operated with a 3.3-V V_{CC}. Refer to Table 4–4 on page 4–11.

(6) The I_{OH} parameter refers to high-level TTL or CMOS output current; the I_{OL} parameter refers to low-level TTL or CMOS output current.

(7) Capacitance is sample-tested only.

Pin Information

Table 4–20 describes EPC1, EPC2, and EPC1441 pin functions during device configuration.

For pin information about enhanced configuration devices, refer to the *Enhanced Configuration Devices (EPC4, EPC8 and EPC16) Data Sheet*.

For pin information about serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64 and EPCS128) Data Sheet*.

Table 4–20. EPC1, EPC2 and EPC1441 Pin Functions During Configuration (Part 1 of 3)

Pin Name	Pin Number			Pin Type	Description
	8-Pin PDIP *(1)*	20-Pin PLCC	32-Pin TQFP *(2)*		
DATA	1	2	31	Output	Serial data output. The DATA pin connects to the DATA0 of the FPGA. DATA is latched into the FPGA on the rising edge of DCLK. The DATA pin is tri-stated before configuration and when the *n*CS pin is high. After configuration, the EPC2 device will drive DATA high, while the EPC1 and EPC1441 device will tri-state DATA.
DCLK	2	4	2	Bidirectional	Clock output when configuring with a single configuration device or when the configuration device is the first (master) device in a chain. Clock input for the next (slave) configuration devices in a chain. The DCLK pin connects to the DCLK of the FPGA. Rising edges on DCLK increment the internal address counter and present the next bit of data on the DATA pin. The counter is incremented only if the OE input is held high, the *n*CS input is held low, and all configuration data has not been transferred to the target device. After configuration or when OE is low, the EPC1, EPC2 and EPC1441 device will drive DCLK low.
OE	3	8	7	Open-Drain Bidirectional	Output enable (active high) and reset (active low). The OE pin connects to the *n*STATUS of the FPGA. A low logic level resets the address counter. A high logic level enables DATA and the address counter to count. If this pin is low (reset) during configuration, the internal oscillator becomes inactive and DCLK drives low. Refer to "Error Detection Circuitry" on page 4–9. The OE pin has an internal programmable 1-kΩ resistor in EPC2 devices. If internal pull-up resistors are use, external pull-up resistors should not be used on these pins. The internal pull-up resistors can be disabled through the **Disable nCS and OE pull-ups on configuration device** option.

Table 4–20. EPC1, EPC2 and EPC1441 Pin Functions During Configuration (Part 2 of 3)

Pin Name	Pin Number			Pin Type	Description
	8-Pin PDIP *(1)*	20-Pin PLCC	32-Pin TQFP *(2)*		
*n*CS	4	9	10	Input	Chip select input (active low). The *n*CS pin connects to the CONF_DONE of the FPGA. A low input allows DCLK to increment the address counter and enables DATA to drive out. If the EPC1 or EPC2 is reset (OE pulled low) while *n*CS is low, the device initializes as the master device in a configuration chain. If the EPC1 or EPC2 device is reset (OE pulled low) while *n*CS is high, the device initializes as a slave device in the chain. The *n*CS pin has an internal programmable 1-kΩ resistor in EPC2 devices. If internal pull-up resistors are use, external pull-up resistors should not be used on these pins.The internal pull-up resistors can be disabled through the **Disable nCS and OE pull-ups on configuration device** option.
*n*CASC	6	12	15	Output	Cascade select output (active low). This output goes low when the address counter has reached its maximum value. When the address counter has reached its maximum value, the configuration device has sent all its configuration data to the FPGA. In a chain of EPC1 or EPC2 devices, the *n*CASC pin of one device is connected to the *n*CS pin of the next device, which permits DCLK to clock data from the next EPC1 or EPC2 device in the chain. For single EPC1 or EPC2 devices and the last device in the chain, *n*CASC is left floating. This pin is only available in EPC1 and EPC2 devices, which support data cascading.
*n*INIT_CONF	N/A	13	16	Open-Drain Output	Allows the INIT_CONF JTAG instruction to initiate configuration. The *n*INIT_CONF pin connects to the *n*CONFIG of the FPGA. If multiple EPC2 devices are used to configure a FPGA, the *n*INIT_CONF of the first EPC2 pin is tied to the FPGA's *n*CONFIG pin, while subsequent devices' *n*INIT_CONF pins are left floating. The INIT_CONF pin has an internal 1-kΩ pull-up resistor that is always active in EPC2 devices. This pin is only available in EPC2 devices.
TDI	N/A	11	13	Input	JTAG data input pin. Connect this pin to V_{CC} if the JTAG circuitry is not used. This pin is only available in EPC2 devices.
TDO	N/A	1	28	Output	JTAG data output pin. Do not connect this pin if the JTAG circuitry is not used. This pin is only available in EPC2 devices.

Table 4–20. EPC1, EPC2 and EPC1441 Pin Functions During Configuration (Part 3 of 3)

Pin Name	Pin Number			Pin Type	Description
	8-Pin PDIP *(1)*	20-Pin PLCC	32-Pin TQFP *(2)*		
TMS	N/A	19	25	Input	JTAG mode select pin. Connect this pin to V_{CC} if the JTAG circuitry is not used. This pin is only available in EPC2 devices.
TCK	N/A	3	32	Input	JTAG clock pin. Connect this pin to ground if the JTAG circuitry is not used. This pin is only available in EPC2 devices.
VCCSEL	N/A	5	3	Input	Mode select for V_{CC} supply. VCCSEL must be connected to ground if the device uses a 5.0-V power supply (V_{CC} = 5.0 V). VCCSEL must be connected to V_{CC} if the device uses a 3.3-V power supply (V_{CC} = 3.3 V). This pin is only available in EPC2 devices.
VPPSEL	N/A	14	17	Input	Mode select for V_{PP}. VPPSEL must be connected to ground if V_{PP} uses a 5.0-V power supply (V_{PP} = 5.0 V). VPPSEL must be connected to V_{CC} if V_{PP} uses a 3.3-V power supply (V_{PP} = 3.3 V). This pin is only available in EPC2 devices.
VPP	N/A	18	23	Power	Programming power pin. For the EPC2 device, this pin is normally tied to V_{CC}. If the EPC2 V_{CC} is 3.3 V, V_{PP} can be tied to 5.0 V to improve in-system programming times. For EPC1 and EPC1441 devices, V_{PP} must be tied to V_{CC}. This pin is only available in EPC2 devices.
VCC	7, 8	20	27	Power	Power pin.
GND	5	10	12	Ground	Ground pin. A 0.2-µF decoupling capacitor must be placed between the VCC and GND pins.

Notes to Table 4–20:

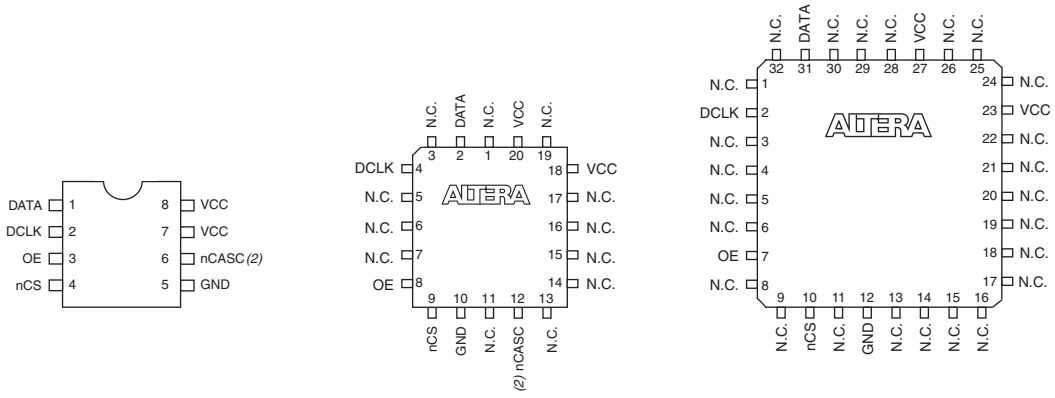
(1) This package is available for EPC1 and EPC1441 devices only.

(2) This package is available for EPC2 and EPC1441 devices only.

Package

Figure 4–6 and Figure 4–7 show the configuration device package pin-outs.

Figure 4–6. EPC1, EPC1064, EPC1064V, EPC1213, and EPC1441 Package Pin-Out Diagrams ***(Note 1)***



8-Pin PDIP

EPC1
EPC1441
EPC1213
EPC1064
EPC1064V

20-Pin PLCC

EPC1
EPC1441
EPC1213
EPC1064
EPC1064V

32-Pin TQFP

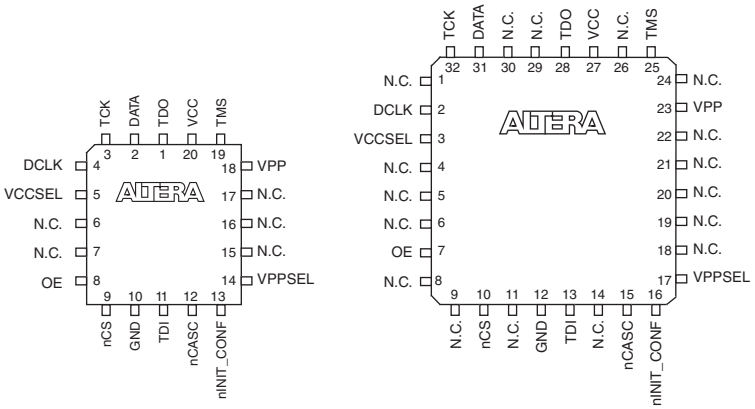
EPC1441
EPC1064
EPC1064V

Notes to Figure 4–6:

(1) EPC1 and EPC1441 devices are one-time programmable devices. ISP is not available in these devices.

(2) The nCASC pin is available on EPC1 devices, which allows them to be cascaded. On the EPC1441 devices, nCASC is a reserved pin and should be left unconnected.

Figure 4–7. EPC2 Package Pin-Out Diagrams



20-Pin PLCC

32-Pin TQFP

For package outlines and drawings, refer to the Altera Device Package Information Data Sheet.

Ordering Codes

Table 4–21. shows the ordering codes for the EPC1, EPC2, and EPC1441 configuration devices.

Table 4–21. Configuration Device Ordering Codes

Device	Package	Temperature	Ordering Code
EPC2	32-pin TQFP	Commercial	EPC2TC32
EPC2	32-pin TQFP	Industrial	EPC2TI32
EPC2	20-pin PLCC	Commercial	EPC2LC20
EPC2	20-pin PLCC	Industrial	EPC2LI20
EPC1	20-pin PLCC	Commercial	EPC1LC20
EPC1	20-pin PLCC	Industrial	EPC1LI20
EPC1	8-pin PDIP	Commercial	EPC1PC8
EPC1	8-pin PDIP	Industrial	EPC1PI8
EPC1441	32-pin TQFP	Commercial	EPC1441TC32
EPC1441	32-pin TQFP	Industrial	EPC1441TI32
EPC1441	20-pin PLCC	Commercial	EPC1441LC20
EPC1441	20-pin PLCC	Industrial	EPC1441LI20
EPC1441	8-pin PDIP	Commercial	EPC1441PC8
EPC1441	8-pin PDIP	Industrial	EPC1441PI8

Document Revision History

Table 4–22 lists the revision history for this chapter.

Table 4–22. Chapter Revision History

Date	Version	Changes
December 2009	2.4	■ Updated "Features" section. ■ Removed "Referenced Documents" section.
October 2008	2.3	■ Updated "Features" and "IEEE Std. 1149.1 (JTAG) Boundary-Scan Testing" sections. ■ Updated Table 5–2 and Table 5–16. ■ Added "Referenced Documents" section. ■ Updated new document format.
April 2007	2.2	Added document revision history.
July 2004	2.0	Added Stratix II and Cyclone II device information throughout chapter.
September 2003	1.0	Initial Release.

5. Quad-Serial Configuration (EPCQ) Devices Datasheet

CF52012-1.0

This chapter describes quad-serial configuration (EPCQ) devices, their features, operations, power mode, timing information, and operating conditions.

Supported Devices

Table 5–1 lists the features of the supported Altera® EPCQ devices and the amount of configuration space available.

Table 5–1. Altera EPCQ Devices

Device	Memory Size (bits)	On-Chip Decompression Support	ISP Support	Cascading Support	Reprogrammable	Operating Voltage (V)
EPCQ128	134,271,728	No	Yes	No	Yes	3.3
EPCQ256	268,435,456	No	Yes	No	Yes	3.3

Features

EPCQ devices offer the following features:

- Serial or quad-serial FPGA configuration in devices that support active serial (AS) x1 or AS x4 configuration schemes
- Low cost, low pin count, and non-volatile memory
- 2.7-V to 3.6-V operation
- Available in 16-pin small-outline integrated circuit (SOIC) package
- Reprogrammable memory with more than 100,000 erase or program cycles
- Write protection support for memory sectors using status register bits
- Fast read, extended dual input fast read, and extended quad input fast read of the entire memory using a single operation code
- Write bytes, extended dual input fast write bytes, and extended quad input fast write bytes of the entire memory using a single operation code
- Reprogrammable with an external microprocessor using the SRunner software driver
- In-system programming (ISP) support with the SRunner software driver
- ISP support with USB-Blaster™, EthernetBlaster II, or EthernetBlaster download cables
- EPCQ devices cannot be cascaded
- By default, the memory array is erased and the bits are set to `1`

© 2011 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

Memory Array Organization

Table 5–2 lists the memory array organization in supported EPCQ devices.

Table 5–2. Memory Array Organization in EPCQ Devices

Details	EPCQ128	EPCQ256
Bytes	16,777,216 bytes [128 megabits (Mb)]	33,554,432 bytes (256 Mb)
Number of sectors	256	512
Bytes per sector	65,536 bytes [512 kilobits (Kb)]	65,536 bytes (512 Kb)
Total number of subsectors	4,096 *(1)*	8,192 *(1)*
Bytes per subsector	4,096 bytes (32 Kb)	4,096 bytes (32 Kb)
Pages per sector	256	256
Total number of pages	65,536	131,072
Bytes per page	256 bytes	256 bytes

Note to Table 5–2:

(1) Every sector is further divided into 16 subsectors with 4 KB of memory. Because of this, there are 4,096 (256 x 16) subsectors for the EPCQ128 device and 8,192 (512 x 16) subsectors for the EPCQ256 device.

Address Range for EPCQ128

Table 5–3 lists the address range for each sector in EPCQ128 devices.

Table 5–3. Address Range for Sectors 255..0 and Subsectors 4095..0 in EPCQ128 Devices —Preliminary (Part 1 of 3)

Sector	Subsector	Address Range (Byte Addresses in HEX)	
		Start	End
255	4095	FFF000	FFFFFF
	4094	FFE000	FFEFFF
	...	. . .	. . .
	4082	FF2000	FF2FFF
	4081	FF1000	FF1FFF
	4080	FF0000	FF0FFF
254	4079	FEF000	FEFFFF
	4078	FEE000	FEEFFF
	...	. . .	. . .
	4066	FE2000	FE2FFF
	4065	FE1000	FE1FFF
	4064	FE0000	FE0FFF

Table 5–3. Address Range for Sectors 255..0 and Subsectors 4095..0 in EPCQ128 Devices —Preliminary (Part 2 of 3)

Sector	Subsector	Address Range (Byte Addresses in HEX)	
		Start	End
129	2079	81F000	81FFFF
	2078	81E000	81EFFF
	...	...	...
	2066	812000	812FFF
	2065	811000	811FFF
	2064	810000	810FFF
128	2063	80F000	80FFFF
	2062	80E000	80EFFF
	...	...	...
	2050	802000	802FFF
	2049	801000	801FFF
	2048	800000	800FFF
127	2047	7FF000	7FFFFF
	2046	7FE000	7FEFFF
	...	...	...
	2034	7F2000	7F2FFF
	2033	7F1000	7F1FFF
	2032	7F0000	7F0FFF
64	1039	40F000	40FFFF
	1038	40E000	40EFFF
	...	...	...
	1026	402000	402FFF
	1025	401000	401FFF
	1024	400000	400FFF
63	1023	3FF000	3FFFFF
	1022	3FE000	3FEFFF
	...	...	...
	1010	3F2000	3F2FFF
	1009	3F1000	3F1FFF
	1008	3F0000	3F0FFF
62	1007	3EF000	3EFFFF
	1006	3EE000	3EEFFF
	...	...	...
	994	3E2000	3E2FFF
	993	3E1000	3E1FFF
	992	3E0000	3E0FFF

Table 5–3. Address Range for Sectors 255..0 and Subsectors 4095..0 in EPCQ128 Devices —Preliminary (Part 3 of 3)

Sector	Subsector	Address Range (Byte Addresses in HEX)	
		Start	End
1	31	1F000	1FFFF
	30	1E000	1EFFF
	...	...	...
	18	12000	12FFF
	17	11000	11FFF
	16	10000	10FFF
0	15	F000	FFFF
	14	E000	EFFF
	...	...	...
	2	2000	2FFF
	1	1000	1FFF
	0	H'0000000	H'0000FFF

Address Range for EPCQ256

Table 5–4 lists the address range for each sector in EPCQ256 devices.

Table 5–4. Address Range for Sectors 511..0 and Subsectors 8191..0 in EPCQ256 Devices —Preliminary (Part 1 of 3)

Sector	Subsector	Address Range (Byte Addresses in HEX)	
		Start	End
511	8191	1FFF000	1FFFFFF
	8190	1FFE000	1FFEFFF
	...	...	...
	8178	1FF2000	1FF2FFF
	8177	1FF1000	1FF1FFF
	8176	1FF0000	1FF0FFF
510	8175	1FEF000	1FEFFFF
	8174	1FEE000	1FEEFFF
	...	...	...
	8162	1FE2000	1FE2FFF
	8161	1FE1000	1FE1FFF
	8160	1FE0000	1FE0FFF

Table 5–4. Address Range for Sectors 511..0 and Subsectors 8191..0 in EPCQ256 Devices —Preliminary (Part 2 of 3)

Sector	Subsector	Address Range (Byte Addresses in HEX)	
		Start	End
257	4127	101F000	101FFFF
	4126	101E000	101EFFF
	...	...	...
	4114	1012000	1012FFF
	4113	1011000	1011FFF
	4112	1010000	1010FFF
256	4111	100F000	100FFFF
	4110	100E000	100EFFF
	...	...	...
	4098	1002000	1002FFF
	4097	1001000	1001FFF
	4096	1000000	1000FFF
255	4095	FFF000	FFFFFF
	4094	FFE000	FFEFFF
	...	...	...
	4082	FF2000	FF2FFF
	4081	FF1000	FF1FFF
	4080	FF0000	FF0FFF
254	4079	FEF000	FEFFFF
	4078	FEE000	FEEFFF
	...	...	...
	4066	FE2000	FE2FFF
	4065	FE1000	FE1FFF
	4064	FE0000	FE0FFF
129	2079	81F000	81FFFF
	2078	81E000	81EFFF
	...	...	...
	2066	812000	812FFF
	2065	811000	811FFF
	2064	810000	810FFF
128	2063	80F000	80FFFF
	2062	80E000	80EFFF
	...	...	...
	2050	802000	802FFF
	2049	801000	801FFF
	2048	800000	800FFF

Table 5–4. Address Range for Sectors 511..0 and Subsectors 8191..0 in EPCQ256 Devices —Preliminary (Part 3 of 3)

Sector	Subsector	Address Range (Byte Addresses in HEX)	
		Start	End
127	2047	7FF000	7FFFFF
	2046	7FE000	7FEFFF
	...	...	...
	2034	7F2000	7F2FFF
	2033	7F1000	7F1FFF
	2032	7F0000	7F0FFF
64	1039	40F000	40FFFF
	1038	40E000	40EFFF
	...	...	...
	1026	402000	402FFF
	1025	401000	401FFF
	1024	400000	400FFF
63	1023	3FF000	3FFFFF
	1022	3FE000	3FEFFF
	...	...	...
	1010	3F2000	3F2FFF
	1009	3F1000	3F1FFF
	1008	3F0000	3F0FFF
62	1007	3EF000	3EFFFF
	1006	3EE000	3EEFFF
	...	...	...
	994	3E2000	3E2FFF
	993	3E1000	3E1FFF
	992	3E0000	3E0FFF
1	31	1F000	1FFFF
	30	1E000	1EFFF
	...	...	...
	18	12000	12FFF
	17	11000	11FFF
	16	10000	10FFF
0	15	F000	FFFF
	14	E000	EFFF
	...	...	...
	2	2000	2FFF
	1	1000	1FFF
	0	H'0000000	H'0000FFF

Memory Operations

This section describes the operations that you can use to access the memory in EPCQ devices. When performing the operation, addresses and data are shifted in and out of the device serially, with the MSB first.

Timing Requirements

When the active low chip select (nCS) signal is driven low, shift in the operation code into the EPCQ device using the serial data (DATA) pin. Each operation code bit is latched into the EPCQ device on the rising edge of the DCLK.

While executing an operation, shift in the desired operation code, followed by the address or data bytes as listed in Table 5–5. The device must drive the nCS pin high after the last bit of the operation sequence is shifted in.

For read operations, the data read is shifted out on the DATA pin. You can drive the nCS pin high when any bit of the data is shifted out.

For write and erase operations, drive the nCS pin high at a byte boundary, that is in a multiple of eight clock pulses. Otherwise, the operation is rejected and not executed.

All attempts to access the memory contents while a write or erase cycle is in progress are rejected, and the write or erase cycle continues unaffected.

Addressing Mode

The 3-byte addressing mode is enabled by default. To access the EPCQ256 memory, you must use the 4-byte addressing mode. In 4-byte addressing mode, the address width is 32-bit address. To enable the 4-byte addressing mode, you must execute the 4BYTEADDREN operation. This addressing mode takes effect immediately after you execute the 4BYTEADDREN operation and remains active in the subsequent power-ups. To disable the 4-byte addressing mode, you must execute the 4BYTEADDREX operation.

If you are using the Quartus® II software or the SRunner software to program the EPCQ256 device, you do not need to execute the 4BYTEADDREN operation. These software automatically enable the 4-byte addressing mode when programming the device.

Summary of Operation Codes

Table 5–5 lists the supported operations.

Table 5–5. Operation Codes for EPCQ Devices—Preliminary

Operation	Operation Code *(1)*	Address Bytes	Dummy Bytes	Data Bytes	DCLK f_{MAX} (MHz) *(2)*
Read status	b'0000 0101	0	0	1 to infinite *(3)*	100
Read bytes	b'0000 0011	3 or 4	0	1 to infinite *(3)*	50
Read device identification	b'1010 101x	0	0	1 to 20 *(3)*	100
Fast read	b'0000 1011	3 or 4	8 *(5)*	1 to infinite *(3)*	100
Extended dual input fast read	b'1011 1011	3 or 4	8 *(5)*	1 to infinite *(3)*	100
Extended quad input fast read	b'1110 1011	3 or 4	10 *(5)*	1 to infinite *(3)*	100
Write enable	b'0000 0110	0	0	0	100
Write disable	b'0000 0100	0	0	0	100
Write status	b'0000 0001	0	0	1	100
Write bytes	b'0000 0010	3 or 4	0	1 to 256 *(4)*	100
Extended dual input fast write bytes	b'1101 0010	3 or 4	0	1 to 256 *(4)*	100
Extended quad input fast write bytes	b'0001 0010	3	0	1 to 256 *(4)*	100
Erase bulk	b'1100 0111	0	0	0	100
Erase sector	b'1101 1000	3 or 4	0	0	100
4BYTEADDREN *(6)*	b'1011 0111	0	0	0	100
4BYTEASSREX *(6)*	b'1110 1001	0	0	0	100

Notes to Table 5–5:

(1) List MSB first and LSB last.

(2) Pending characterization data.

(3) The status register, data, or read device identification is read out at least once on the DATA1 pin and is continuously read out until the nCS pin is driven high.

(4) A write bytes operation requires at least one data byte on the DATA1 pin. If more than 256 bytes are sent to the device, only the last 256 bytes are written to the memory.

(5) You can configure the number of dummy bytes.

(6) This operation is applicable for EPCQ256 device only.

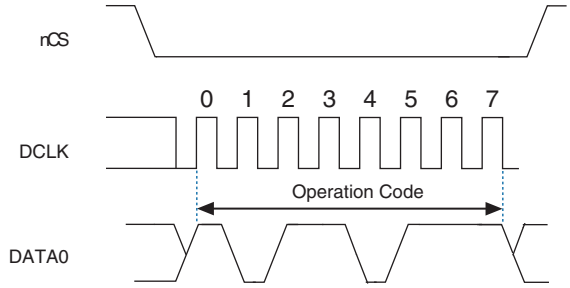
4BYTEADDREN AND 4BYTEADDREX Operations

To enable 4BYTEADDREN or 4BYTEADDREX operations, you can select the device by driving the nCS signal low, followed by shifting in the operation code through DATA0.

You must execute a write enable operation before you can enable the 4BYTEADDREN or 4BYTEADDREX operations. For more information, refer to "Write Enable Operation".

Figure 5–1 shows the timing diagram for the 4BYTEADDREN and 4BYTEADDREX operations.

Figure 5–1. 4BYTEADDREN and 4BYTEADDREX Timing Diagram

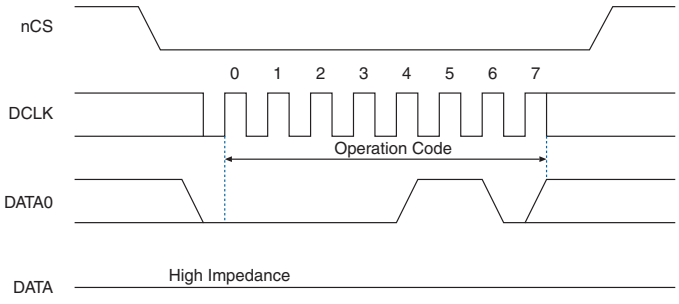


Write Enable Operation

When you enable the write enable operation, the write enable latch bit is set to 1 in the status register. You must execute this operation before the write bytes, write status, erase bulk, erase sector, extended dual input fast write bytes, extended quad input fast write bytes, 4BYTEADDREN, and 4BYTEADDREX operations.

Figure 5–2 shows the timing diagram for the write enable operation.

Figure 5–2. Write Enable Operation Timing Diagram



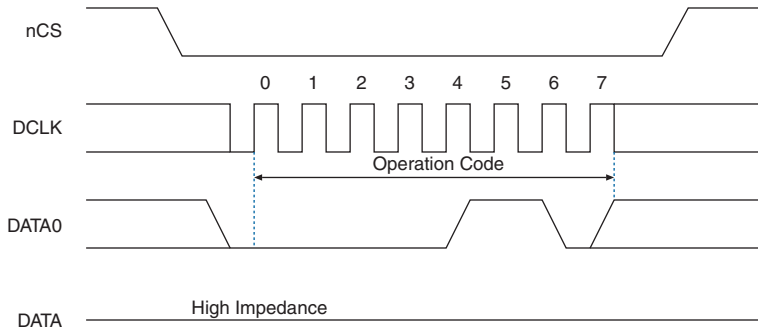
Write Disable Operation

The write disable operation resets the write enable latch bit in the status register. To prevent the memory from being written unintentionally, the write enable latch bit is automatically reset when implementing the write disable operation, and under the following conditions:

- Power up
- Write bytes operation completion
- Write status operation completion
- Erase bulk operation completion
- Erase sector operation completion
- Extended dual input fast write bytes operation completion
- Extended quad input fast write bytes operation completion

Figure 5–3 shows the timing diagram for the write disable operation.

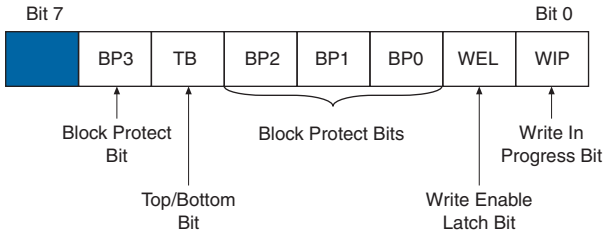
Figure 5–3. Write Disable Operation Timing Diagram



Read Status Operation

You can use the read status operation to read the status register. Figure 5–4 shows the status bits in the status register of the EPCQ devices.

Figure 5–4. EPCQ128 and EPCQ256 Status Register Status Bits



If you set the write in progress bit to 1, the EPCQ device executes a write or erase cycle and 0 indicates that no write or erase cycle is in progress.

If you set the write enable latch bit to 0, the EPCQ device rejects a write or erase cycle. You must set the write enable latch bit to 1 before every write bytes, write status, erase bulk, and erase sector operations.

Use the top or bottom bit (TB bit) with the block protect bits to determine that the protected area starts from the top or bottom of the memory array. When the top or bottom bit is set to 0, the protected area starts from the top of the memory array. When the top or bottom bit is set to 1, the protected area starts from the bottom of the memory array.

The non-volatile block protect bits determine the area of the memory protected from being written or erased unintentionally. Table 5–6 through Table 5–9 on page 5–13 list the protected area in EPCQ128 and EPCQ256 devices with reference to the block protect bits. The erase bulk operation is only available when all the block protect bits are set to 0. When any of the block protect bits are set to 1, the relevant area is protected from being written by a write bytes operation or erased by an erase sector operation.

Table 5–6. Block Protection Bits in EPCQ128 when TB Bit is Set to 0

Status Register Content					Memory Content	
TB Bit	BP3 Bit	BP2 Bit	BP1 Bit	BP0 Bit	Protected Area	Unprotected Area
0	0	0	0	0	None	All sectors (sectors 0 to 255)
0	0	0	0	1	Upper 256	Sectors 0 to 254
0	0	0	1	0	Upper 128	Sectors 0 to 253
0	0	0	1	1	Upper 64	Sectors 0 to 251
0	0	1	0	0	Upper 32	Sectors 0 to 247
0	0	1	0	1	Upper 16	Sectors 0 to 239
0	0	1	1	0	Upper 8	Sectors 0 to 223
0	0	1	1	1	Upper quarter	Sectors 0 to 191
0	1	0	0	0	Upper half (sector 255)	Lower half (sectors 0 to 127)
0	1	0	0	1	All sectors	None
0	1	0	1	0	All sectors	None
0	1	0	1	1	All sectors	None
0	1	1	0	0	All sectors	None
0	1	1	0	1	All sectors	None
0	1	1	1	0	All sectors	None
0	1	1	1	1	All sectors (128 Mb, 256 sectors)	None

Table 5–7. Block Protection Bits in EPCQ128 when TB Bit is Set to 1

Status Register Content					Memory Content	
TB Bit	BP3 Bit	BP2 Bit	BP1 Bit	BP0 Bit	Protected Area	Unprotected Area
1	0	0	0	0	None	All sectors (sectors 0 to 255)
1	0	0	0	1	Lower 256	Sectors 1 to 255
1	0	0	1	0	Lower 128	Sectors 2 to 255
1	0	0	1	1	Lower 64	Sectors 4 to 255
1	0	1	0	0	Lower 32	Sectors 8 to 255
1	0	1	0	1	Lower 16	Sectors 16 to 255
1	0	1	1	0	Lower 8	Sectors 32 to 255
1	0	1	1	1	Lower quarter	Sectors 64 to 255
1	1	0	0	0	Lower half	Sectors 128 to 255
1	1	0	0	1	All sectors	None
1	1	0	1	0	All sectors	None
1	1	0	1	1	All sectors	None
1	1	1	0	0	All sectors	None
1	1	1	0	1	All sectors	None
1	1	1	1	0	All sectors	None
1	1	1	1	1	All sectors	None

Table 5–8. Block Protection Bits in EPCQ256 when TB Bit is Set to 0

Status Register Content					Memory Content	
TB Bit	BP3 Bit	BP2 Bit	BP1 Bit	BP0 Bit	Protected Area	Unprotected Area
0	0	0	0	0	None	All sectors
0	0	0	0	1	Upper 512	Sectors 0 to 511
0	0	0	1	0	Upper 256	Sectors 0 to 510
0	0	0	1	1	Upper 128	Sectors 0 to 508
0	0	1	0	0	Upper 64	Sectors 0 to 504
0	0	1	0	1	Upper 32	Sectors 0 to 496
0	0	1	1	0	Upper 16	Sectors 0 to 480
0	0	1	1	1	Upper 8	Sectors 0 to 448
0	1	0	0	0	Upper quarter	Sectors 0 to 384
0	1	0	0	1	Upper half	Sectors 0 to 256
0	1	0	1	0	All sectors	None
0	1	0	1	1	All sectors	None
0	1	1	0	0	All sectors	None
0	1	1	0	1	All sectors	None
0	1	1	1	0	All sectors	None
0	1	1	1	1	All sectors	None

Table 5–9. Block Protection Bits in EPCQ256 when TB Bit is Set to 1

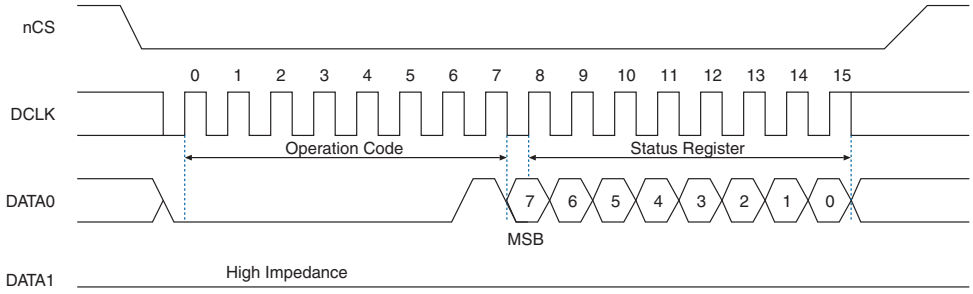
Status Register Content					Memory Content	
TB Bit	BP3 Bit	BP2 Bit	BP1 Bit	BP0 Bit	Protected Area	Unprotected Area
1	0	0	0	0	None	All sectors
1	0	0	0	1	Lower 512	Sectors 1 to 511
1	0	0	1	0	Lower 256	Sectors 2 to 511
1	0	0	1	1	Lower 128	Sectors 4 to 511
1	0	1	0	0	Lower 64	Sectors 8 to 511
1	0	1	0	1	Lower 32	Sectors 16 to 511
1	0	1	1	0	Lower 16	Sectors 32 to 511
1	0	1	1	1	Lower 8	Sectors 64 to 511
1	1	0	0	0	Lower quarter	Sectors 128 to 511
1	1	0	0	1	Lower half	Sectors 256 to 511
1	1	0	1	0	All sectors	None
1	1	0	1	1	All sectors	None
1	1	1	0	0	All sectors	None
1	1	1	0	1	All sectors	None
1	1	1	1	0	All sectors	None
1	1	1	1	1	All sectors	None

Write Status Operation

The write status operation does not affect the write enable latch and write in progress bits. You can use the write status operation to set the status register block protection and top or bottom bits. Therefore, you can implement this operation to protect certain memory sectors. For more information, refer to Table 5–6 on page 5–12 through Table 5–9. After setting the block protect bits, the protected memory sectors are treated as read-only memory. You must execute the write enable operation before the write status operation.

Figure 5–5 shows the timing diagram for the write status operation.

Figure 5–5. Write Status Operation Timing Diagram



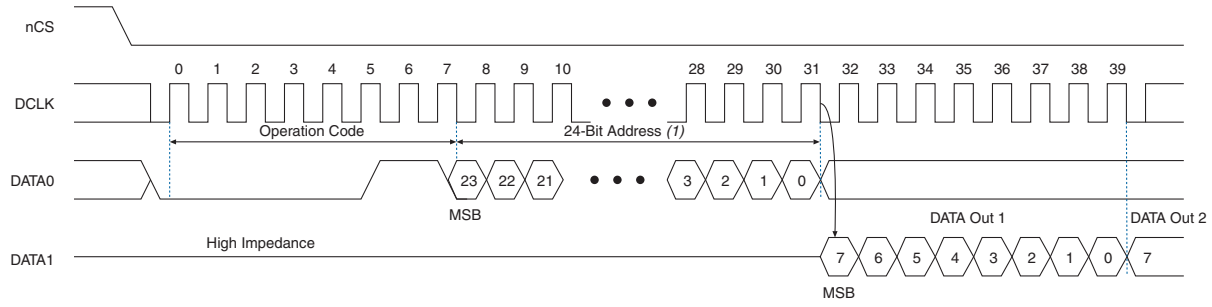
Immediately after the nCS signal drives high, the device initiates the self-timed write status cycle. The self-timed write status cycle usually takes 5 ms for all EPCQ devices and is guaranteed to be less than 8 ms. For more information, refer to t_{WS} in Table 5–11 on page 5–25. You must account for this delay to ensure that the status register is written with the desired block protect bits. Alternatively, you can check the write in progress bit in the status register by executing the read status operation while the self-timed write status cycle is in progress. Set the write in progress bit to 1 during the self-timed write status cycle and 0 when it is complete.

Read Bytes Operation

When you execute the read bytes operation, you first shift in the read bytes operation code, followed by a 3-byte addressing mode (A[23..0]) or a 4-byte addressing mode (A[31..0]). Each address bit must be latched in on the rising edge of the DCLK signal. After the address is latched in, the memory contents of the specified address are shifted out serially on the DATA1 pin, beginning with the MSB. For reading Raw Programming Data File (**.rpd**), the content is shifted out serially beginning with the LSB. Each data bit is shifted out on the falling edge of the DCLK signal. The maximum DCLK frequency during the read bytes operation is 20 MHz.

Figure 5–6 shows the timing diagram for the read bytes operation.

Figure 5–6. Read Bytes Operation Timing Diagram



Note to Figure 5–6:

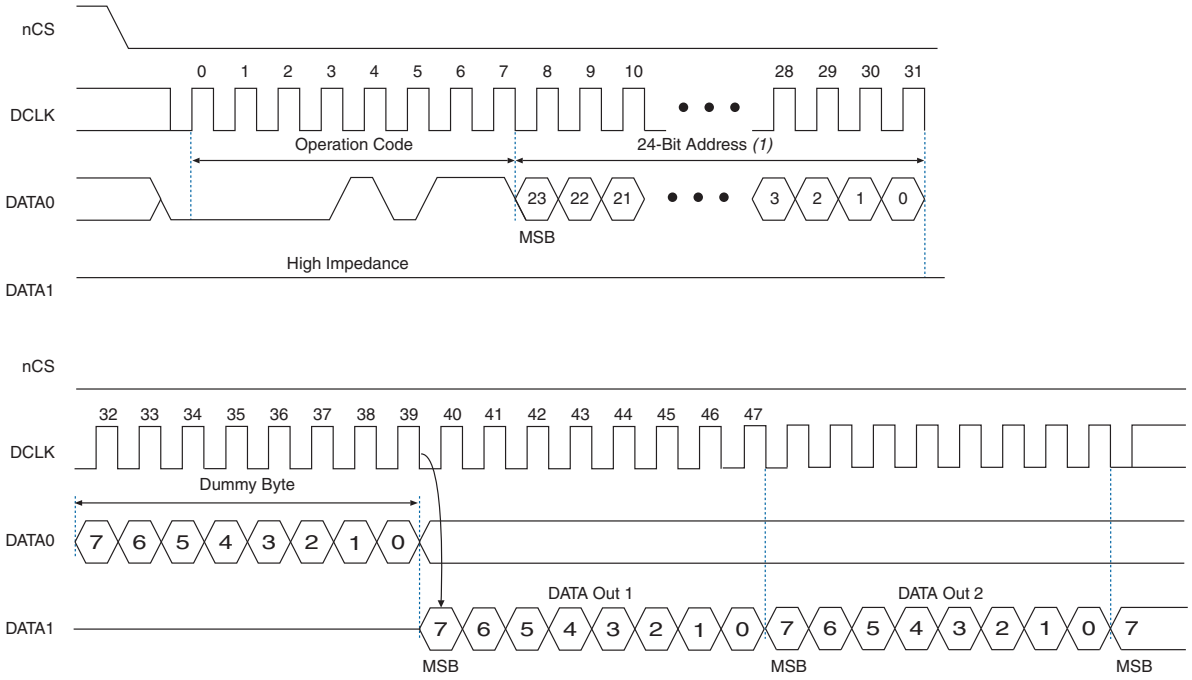
(1) To access the entire EPCQ256 memory, use 4-byte addressing mode. In the 4-byte addressing mode, the address width is 32-bit address.

The first byte address can be at any location. The device automatically increases the address to the next higher address after shifting out each byte of data. Therefore, the device can read the whole memory with a single read bytes operation. When the device reaches the highest address, the address counter restarts at 0x000000, allowing the memory contents to be read out indefinitely until the read bytes operation is terminated by driving the nCS signal high. If the read bytes operation is shifted in while a write or erase cycle is in progress, the operation is not executed and does not affect the write or erase cycle in progress.

Fast Read Operation

When you execute the fast read operation, you first shift in the fast read operation code, followed by a 3-byte addressing mode (A[23..0]) or a 4-byte addressing mode (A[31..0]), and a dummy byte with each bit being latched-in during the rising edge of the DCLK signal. Then, the memory contents at that address is shifted out on DATA1 with each bit being shifted out at a maximum frequency of 50 MHz during the falling edge of the DCLK signal. Figure 5–7 shows the operation sequence of the fast read operation.

Figure 5–7. Fast Read Operation Timing Diagram



Note to Figure 5–7:

(1) To access the entire EPCQ256 memory, use 4-byte addressing mode. In the 4-byte addressing mode, the address width is 32-bit address.

The first byte address can be at any location. The device automatically increases the address to the next higher address after shifting out each byte of data. Therefore, the device can read the whole memory with a single fast read operation. When the device reaches the highest address, the address counter restarts at 0x000000, allowing the read sequence to continue indefinitely.

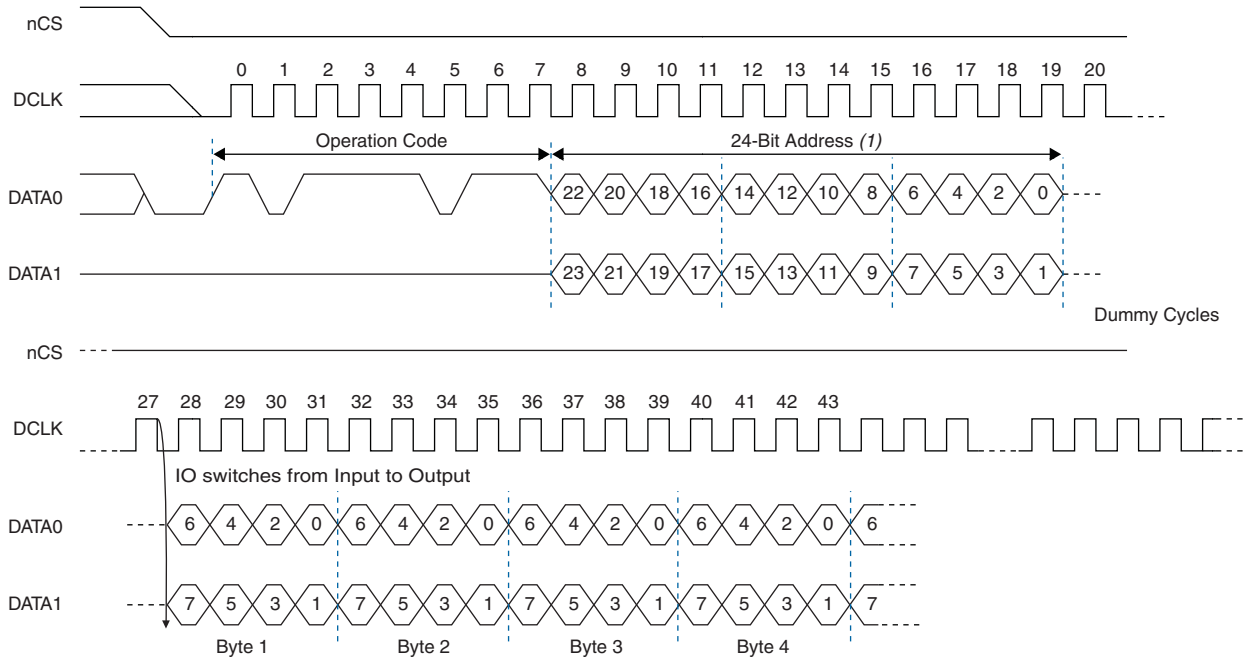
You can terminate the fast read operation by driving the nCS signal high at any time during data output. If the fast read operation is shifted in while an erase, program, or write cycle is in progress, the operation is not executed and does not affect the erase, program, or write cycle in progress.

Extended Dual Input Fast Read Operation

This operation is similar to the fast read operation except that the data and addresses are shifted in and out on the DATA0 and DATA1 pins.

Figure 5–8 shows the operation sequence of the extended dual input fast read operation.

Figure 5–8. Extended Dual Input Fast Read Operation Timing Diagram



Note to Figure 5–8:

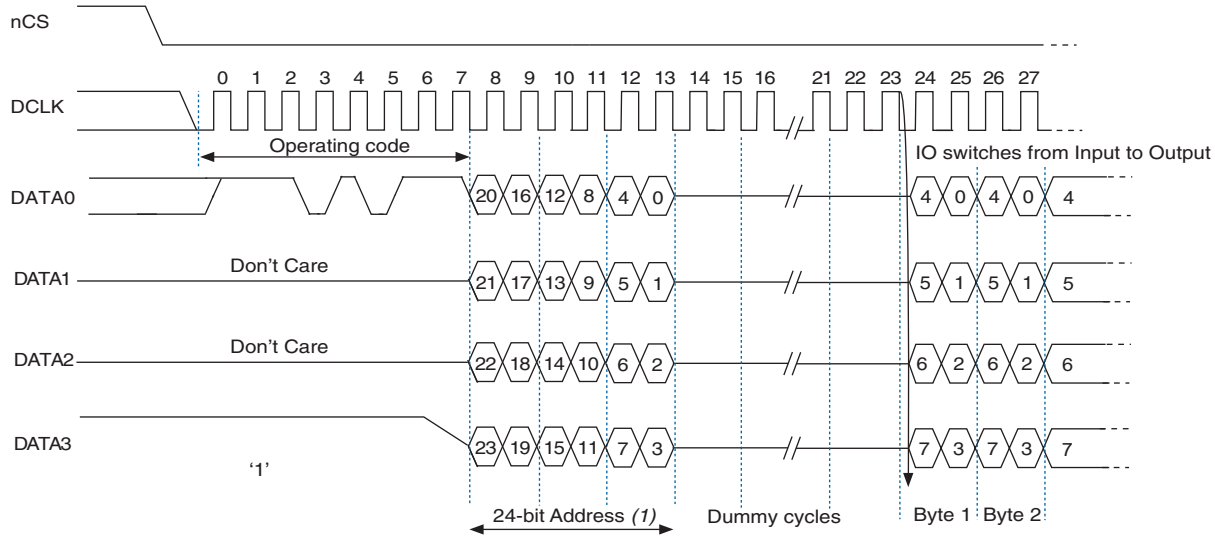
(1) To access the entire EPCQ256 memory, use 4-byte addressing mode. In the 4-byte addressing mode, the address width is 32-bit address.

Extended Quad Input Fast Read Operation

This operation is similar to the extended dual input fast read operation except that the data and addresses are shifted in and out on the DATA0, DATA1, DATA2, and DATA3 pins.

Figure 5–9 shows the operation sequence of the extended quad input fast read operation.

Figure 5–9. Extended Quad Input Fast Read Operation Timing Diagram


nCS
DCLK
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 21 22 23 24 25 26 27
Operating code
IO switches from Input to Output
DATA0
20 16 12 8 4 0
4 0 4 0 4
DATA1
Don't Care
21 17 13 9 5 1
5 1 5 1 5
DATA2
Don't Care
22 18 14 10 6 2
6 2 6 2 6
DATA3
'1'
23 19 15 11 7 3
7 3 7 3 7
24-bit Address (1)
Dummy cycles
Byte 1
Byte 2

Note to Figure 5–9:

(1) To access the entire EPCQ256 memory, use 4-byte addressing mode. In the 4-byte addressing mode, the address width is 32-bit address.

Read Device Identification Operation

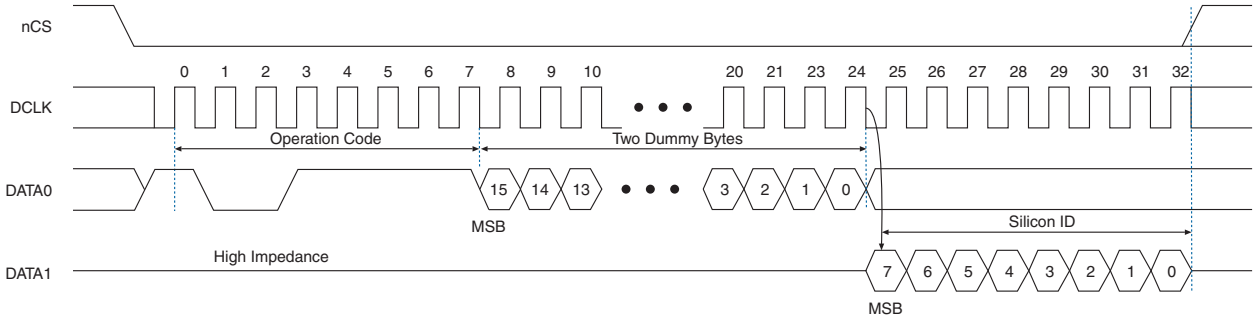
This operation reads the 8-bit device identification of the EPCQ device from the DATA1 output pin. If this operation is shifted in while an erase or write cycle is in progress, the operation is not executed and does not affect the erase or write cycle in progress. Table 5–10 lists the EPCQ device identifications.

Table 5–10. EPCQ Device Identification

EPCQ Device	Silicon ID (Binary Value)
EPCQ128	TBD
EPCQ256	TBD

The 8-bit device identification of the EPCQ device is shifted out on the DATA1 pin on the falling edge of the DCLK signal. Figure 5–10 shows the operation sequence of the read device identification operation.

Figure 5–10. Read Device Identification Operation Timing Diagram


nCS
DCLK
DATA0
DATA1
0 1 2 3 4 5 6 7 8 9 10 20 21 23 24 25 26 27 28 29 30 31 32
Operation Code
Two Dummy Bytes
15 14 13 3 2 1 0
MSB
Silicon ID
High Impedance
7 6 5 4 3 2 1 0
MSB

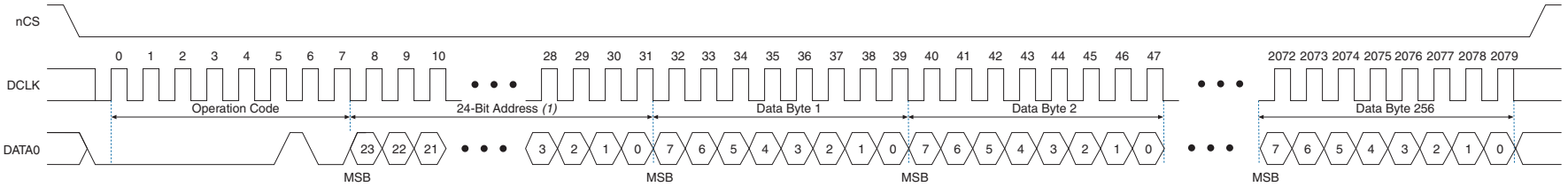
Write Bytes Operation

This operation allows bytes to be written to the memory. You must execute the write enable operation before the write bytes operation. After the write bytes operation is completed, the write enable latch bit in the status register is set to 0.

When you execute the write bytes operation, you shift in the write bytes operation code, followed by a 3-byte addressing mode (A[23..0]) or a 4-byte addressing mode (A[31..0]), and at least one data byte on the DATA0 pin. If the eight LSBs (A[7..0]) are not all 0, all sent data that goes beyond the end of the current page is not written into the next page. Instead, this data is written at the start address of the same page. You must ensure the nCS signal is set low during the entire write bytes operation.

Figure 5–11 shows the operation sequence of the write bytes operation.

Figure 5–11. Write Bytes Operation Timing Diagram



Note to Figure 5–11:

(1) To access the entire EPCQ256 memory, use 4-byte addressing mode. In the 4-byte addressing mode, the address width is 32-bit address.

If more than 256 data bytes are shifted into the EPCQ device with a write bytes operation, the previously latched data is discarded and the last 256 bytes are written to the page. However, if less than 256 data bytes are shifted into the EPCQ device, they are guaranteed to be written at the specified addresses and the other bytes of the same page are not affected.

The device initiates a self-timed write cycle immediately after the nCS signal is driven high. For more information about the self-timed write cycle time, refer to t_{WB} in Table 5–11 on page 5–25. You must account for this amount of delay before another page of memory is written. Alternatively, you can check the write in progress bit in the status register by executing the read status operation while the self-timed write cycle is in progress. The write in progress bit is set to 1 during the self-timed write cycle and 0 when it is complete.

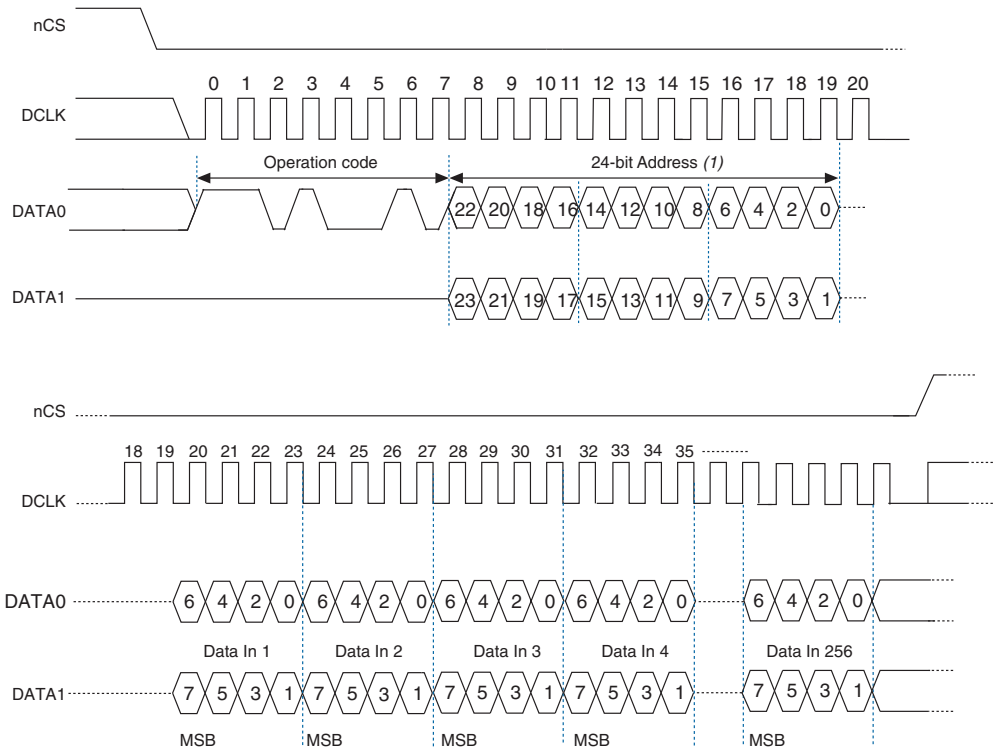
You must erase all the memory bytes of EPCQ devices before you implement the write bytes operation. You can erase all the memory bytes by executing the erase sector operation in a sector or the erase bulk operation throughout the entire memory.

Extended Dual Input Fast Write Bytes Operation

This operation is similar to the write bytes operation except that the data and addresses are shifted in on the DATA0 and DATA1 pins.

Figure 5–12 shows the operation sequence of the extended dual input fast write bytes operation.

Figure 5–12. Extended Dual Input Fast Write Bytes Timing Diagram



Note to Figure 5–12:

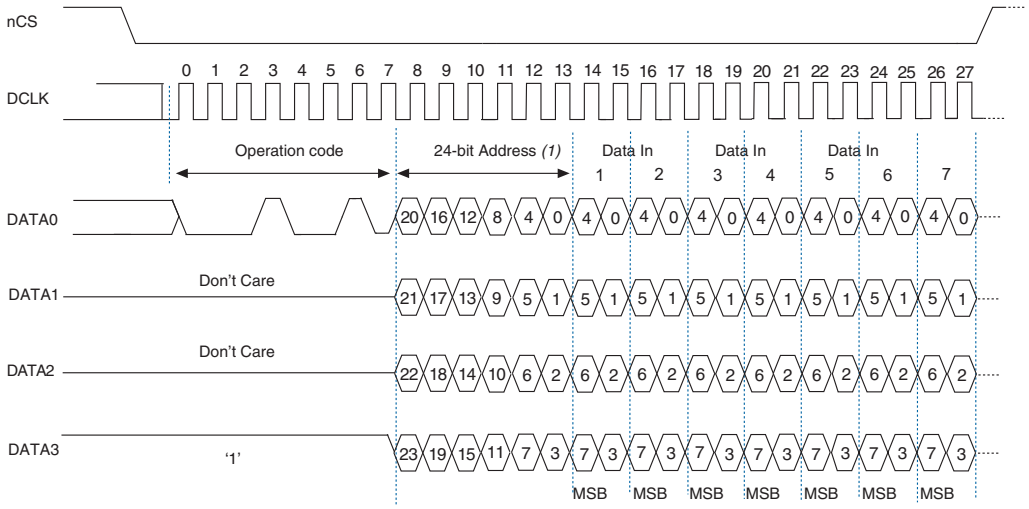
(1) To access the entire EPCQ256 memory, use 4-byte addressing mode. In the 4-byte addressing mode, the address width is 32-bit address.

Extended Quad Input Fast Write Bytes Operation

This operation is similar to the extended dual input fast write bytes operation except that the data and addresses are shifted in on the DATA0, DATA1, DATA2, and DATA3 pins.

Figure 5–13 shows the operation sequence of the extended quad input fast write bytes operation.

Figure 5–13. Extended Quad Input Fast Write Bytes Operation Timing Diagram



Note to Figure 5–13:

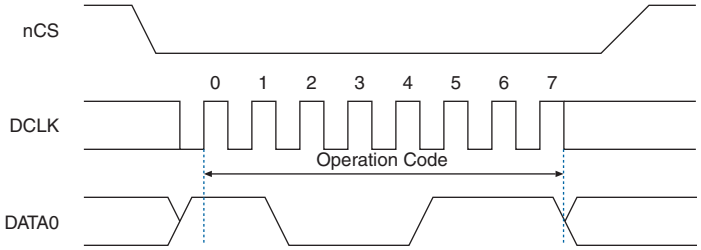
(1) To access the entire EPCQ256 memory, use 4-byte addressing mode. In the 4-byte addressing mode, the address width is 32-bit address.

Erase Bulk Operation

This operation sets all the memory bits to `1` or `0xFF`. Similar to the write bytes operation, you must execute the write enable operation before the erase bulk operation.

You can implement the erase bulk operation by driving the `nCS` signal low and then shifting in the erase bulk operation code on the `DATA0` pin. The `nCS` signal must be driven high after the eighth bit of the erase bulk operation code has been latched in. Figure 5–14 shows the erase bulk operation.

Figure 5–14. Erase Bulk Operation Timing Diagram



The device initiates a self-timed erase bulk cycle immediately after the `nCS` signal is driven high. For more information about the self-timed erase bulk cycle time, refer to t_{WB} in Table 5–11 on page 5–25.

You must account for this delay before accessing the memory contents. Alternatively, you can check the write in progress bit in the status register by executing the read status operation while the self-timed erase cycle is in progress. The write in progress bit is set to `1` during the self-timed erase cycle and `0` when it is complete. The write enable latch bit in the status register is reset to `0` before the erase cycle is complete.

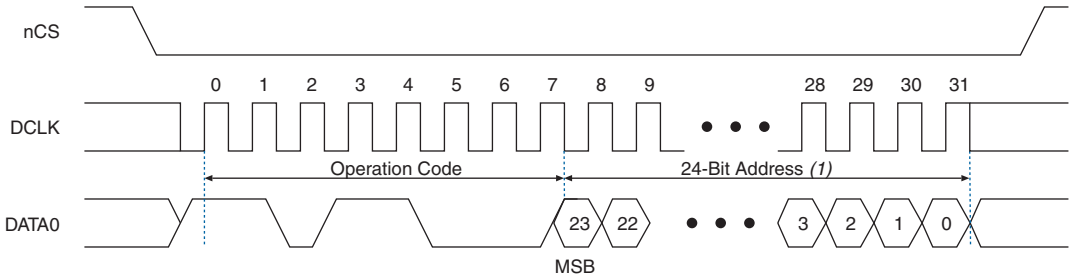
Erase Sector Operation

The erase sector operation allows you to erase a certain sector in the EPCQ device by setting all the bits inside the sector to 1 or 0xFF. This operation is useful if you want to access the unused sectors as a general purpose memory in your applications. You must execute the write enable operation before the erase sector operation.

When you execute the erase sector operation, you must first shift in the erase sector operation code, followed by the 3-byte addressing mode (A[23..0]) or the 4-byte addressing mode (A[31..0]) of the chosen sector on the DATA0 pin. The 3-byte addressing mode or the 4-byte addressing mode for the erase sector operation can be any address inside the specified sector. For more information about the sector address range, refer to Table 5–3 on page 5–2 and Table 5–4 on page 5–4. Drive the nCS signal high after the eighth bit of the erase sector operation code has been latched in.

Figure 5–15 shows the erase sector operation.

Figure 5–15. Erase Sector Operation Timing Diagram



Note to Figure 5–15:

(1) To access the entire EPCQ256 memory, use 4-byte addressing mode. In the 4-byte addressing mode, the address width is 32-bit address.

The device initiates a self-timed erase sector cycle immediately after the nCS signal is driven high. For more information about the self-timed erase sector cycle time, refer to t_{ES} in Table 5–11 on page 5–25. You must account for this amount of delay before another page of memory is written. Alternatively, you can check the write in progress bit in the status register by executing the read status operation while the self-timed erase cycle is in progress. The write in progress bit is set to 1 during the self-timed erase cycle and 0 when it is complete. The write enable latch bit in the status register is set to 0 before the self-timed erase cycle is complete.

Power Mode

EPCQ devices support active and standby power modes. When the nCS signal is low, the device is enabled and is in active power mode. The FPGA is configured while the EPCQ device is in active power mode. When the nCS signal is high, the device is disabled but remains in active power mode until all internal cycles are completed, such as write or erase operations. The EPCQ device then goes into standby power mode. The I_{CC1} and I_{CC0} parameters list the V_{CC} supply current when the device is in active and standby power modes. For more information, refer to Table 5–16 on page 5–28.

Timing Information

Figure 5–16 shows the timing waveform for the write operation.

Figure 5–16. Write Operation Timing Diagram

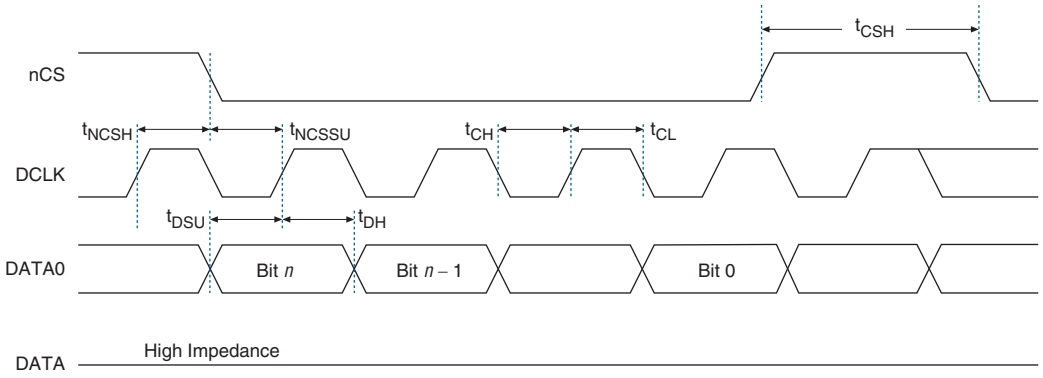


Table 5–11 lists the EPCQ device timing parameters for the write operation.

Table 5–11. Write Operation Parameters—Preliminary

Symbol	Parameter	Min	Typical	Max	Unit
f_{WCLK}	Write clock frequency (from the FPGA, download cable, or embedded processor) for write enable, write disable, read status, read device identification, write bytes, erase bulk, and erase sector operations	—	—	100	MHz
t_{CH} *(1)*	DCLK high time	4	—	—	ns
t_{CL} *(1)*	DCLK low time	4	—	—	ns
t_{NCSSU}	Chip select (nCS) setup time	4	—	—	ns
t_{NCSH}	Chip select (nCS) hold time	4	—	—	ns
t_{DSU}	DATA[] in setup time before the rising edge on DCLK	2	—	—	ns
t_{DH}	DATA[] hold time after the rising edge on DCLK	3	—	—	ns
t_{CSH}	Chip select (nCS) high time	50	—	—	ns
t_{WB} *(2)*	Write bytes cycle time	—	0.6	5	ms
t_{WS} *(2)*	Write status cycle time	—	1.3	8	ms
t_{EB} *(2)*	Erase bulk cycle time for EPCQ128	—	170	250	s
	Erase bulk cycle time for EPCQ256	—	240	480	s
t_{ES} *(2)*	Erase sector cycle time for EPCQ128	—	0.6	3	s
	Erase sector cycle time for EPCQ256				
t_{ESS} *(2)*	Erase subsector cycle time for EPCQ128	—	0.2	2	s
	Erase subsector cycle time for EPCQ256	—	0.3	3	s

Note to Table 5–11:

(1) The value must be larger than or equal to $1/f_{WCLK}$.

(2) Figure 5–16 does not show these parameters.

Figure 5–17 shows the timing waveform for the read operation.

Figure 5–17. Read Operation Timing Diagram

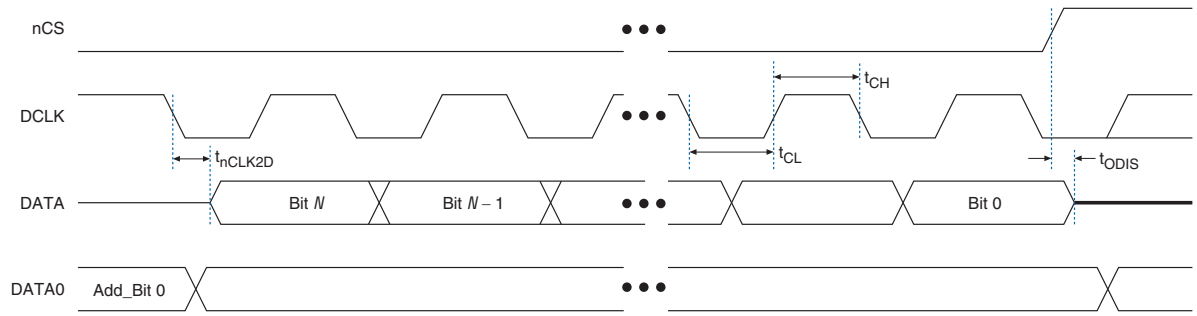


Table 5–12 lists the EPCQ device timing parameters for the read operation.

Table 5–12. Read Operation Parameters—Preliminary

Symbol	Parameter	Min	Max	Unit
f_{RCLK}	Read clock frequency (from the FPGA or embedded processor) for read bytes operations	—	50	MHz
t_{CH}	DCLK high time	4	—	ns
t_{CL}	DCLK low time	4	—	ns
t_{ODIS}	Output disable time after read	—	8	ns
t_{nCLK2D}	Clock falling edge to DATA	—	7	ns

Programming and Configuration File Support

The Quartus II software provides programming support for EPCQ devices. When you select an EPCQ device, the Quartus II software automatically generates the Programmer Object File (**.pof**) to program the device. The software allows you to select the appropriate EPCQ device density that most efficiently stores the configuration data for the selected FPGA.

You can program the EPCQ device in-system by an external microprocessor using the SRunner software driver. The SRunner software driver is developed for embedded EPCQ device programming that you can customize to fit in different embedded systems. The SRunner software driver reads **.rpd** files and writes to the EPCQ devices. The programming time is comparable to the Quartus II software programming time. Because the FPGA reads the LSB of the **.rpd** data first during the configuration process, the LSB of **.rpd** bytes must be shifted out first during the read bytes operation and shifted in first during the write bytes operation.

Writing and reading the **.rpd** file to and from the EPCQ device is different from the other data and address bytes.

During the ISP of an EPCQ device using the USB-Blaster, EthernetBlaster II, or EthernetBlaster download cable, the cable pulls the nCONFIG signal low to reset the FPGA and overrides the 10-kΩ pull-down resistor on the nCE pin of the FPGA. The download cable then uses the interface pins depending on the selected AS mode to program the EPCQ device. When programming is complete, the download cable

releases the interface pins of the EPCQ device and the nCE pin of the FPGA and pulses the nCONFIG signal to start the configuration process.

The FPGA can program the EPCQ device in-system using the JTAG interface with the serial flash loader (SFL). This solution allows you to indirectly program the EPCQ device using the same JTAG interface that is used to configure the FPGA.

For more information about configuration, refer to the configuration chapter in the appropriate device handbook.

For more information about SFL, refer to *AN 370: Using the Serial FlashLoader with the Quartus II software*.

For more information about programming and configuration support, refer to the following documents:

- *USB-Blaster Download Cable User Guide*
- *EthernetBlaster II Communications Cable User Guide*
- *EthernetBlaster Communications Cable User Guide*

Operating Conditions

Table 5–13 through Table 5–17 list information about the absolute maximum ratings, recommended operating conditions, DC operating conditions, I_{CC} supply current, and capacitance for EPCQ devices.

Table 5–13. Absolute Maximum Ratings—Preliminary

Symbol	Parameter	Condition	Min	Max	Unit
V_{CC}	Supply voltage	With respect to GND	–0.6	4.0	V
V_I	DC input voltage	With respect to GND	–0.6	4.0	V
I_{MAX}	DC V_{CC} or GND current	—	—	20	mA
I_{OUT}	DC output current per pin	—	–25	25	mA
P_D	Power dissipation	—	—	54	mW
T_{STG}	Storage temperature	No bias	–65	150	°C
T_{AMB}	Ambient temperature	Under bias	–65	135	°C
T_J	Junction temperature	Under bias	—	135	°C

Table 5–14. Recommended Operating Conditions—Preliminary (Part 1 of 2)

Symbol	Parameter	Condition	Min	Max	Unit
V_{CC}	Supply voltage	*(1)*	2.7	3.6	V
V_I	Input voltage	With respect to GND	–0.5	0.4 + V_{CC}	V
V_O	Output voltage	—	0	V_{CC}	V

Table 5–14. Recommended Operating Conditions—Preliminary (Part 2 of 2)

Symbol	Parameter	Condition	Min	Max	Unit
T_A	Operating temperature	For commercial use	0	70	°C
		For industrial use	–40	85	°C
		For extended industrial temperature use	–40	125	°C
t_R	Input rise time	—	—	5	ns
t_F	Input fall time	—	—	5	ns

Note to Table 5–14:

(1) The maximum V_{CC} rise time is 100 ms.

Table 5–15. DC Operating Conditions—Preliminary

Symbol	Parameter	Condition	Min	Max	Unit
V_{IH}	High-level input voltage	—	0.7 x V_{CC}	V_{CC} + 0.4	V
V_{IL}	Low-level input voltage	—	–0.5	0.3 x V_{CC}	V
V_{OH}	High-level output voltage	I_{OH} = –100 µA *(1)*	V_{CC} - 0.2	—	V
V_{OL}	Low-level output voltage	I_{OL} = 1.6 mA *(2)*	—	0.4	V
I_I	Input leakage current	V_I =V_{CC} or GND	–10	10	µA
I_{OZ}	Tri-state output off-state current	V_O = V_{CC} or GND	–10	10	µA

Notes to Table 5–15:

(1) The I_{OH} parameter refers to the high-level TTL or CMOS output current.

(2) The I_{OL} parameter refers to the low-level TTL or CMOS output current.

Table 5–16. I_{CC} Supply Current—Preliminary

Symbol	Parameter	Condition	Min	Max	Unit
I_{CC0}	V_{CC} supply current (standby)	—	—	100	µA
I_{CC1}	V_{CC} supply current (during active power mode)	—	5	20	mA

Table 5–17. Capacitance *(Note 1)*—Preliminary

Symbol	Parameter	Condition	Min	Max	Unit
C_{IN}	Input pin capacitance	V_{IN} = 0 V	—	6	pF
C_{OUT}	Output pin capacitance	V_{OUT} = 0 V	—	8	pF

Note to Table 5–17:

(1) Capacitance is sample-tested only at T_A = 25 x C and at a 20-MHz frequency.

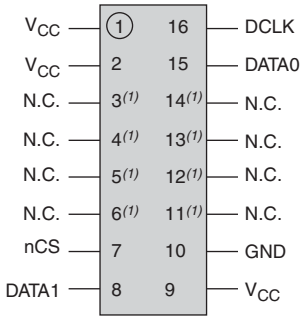
Pin Information

Figure 5–18 and Figure 5–19 show the EPCQ device in a 16-pin device. The following lists the control pins on the EPCQ device:

- Serial data 3 (DATA3)
- Serial data 2 (DATA2)
- Serial data 1 (DATA1)
- Serial data 0 (DATA0)
- Serial clock (DCLK)
- Chip select (nCS)

For more information about configuration, refer to the configuration chapter in the appropriate device handbook.

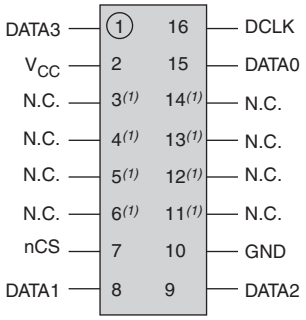
Figure 5–18. EPCQ AS x1 Pin-Out Diagram



Note to Figure 5–18:

(1) These pins are left floating or you can connect the pins to V_{CC} or GND, whichever is more convenient for your board.

Figure 5–19. EPCQ AS x4 Pin-Out Diagram



Note to Figure 5–19:

(1) These pins are left floating or you can connect the pins to V_{CC} or GND, whichever is more convenient for your board.

Table 5–18 lists the pin description of the EPCQ device.

Table 5–18. EPCQ Device Pin Description—Preliminary (Part 1 of 2)

Pin Name	Pin Number in 16-Pin SOIC Package	Pin Type	Description
DATA1	8	I/O	For AS x1 and x4 modes, use this pin as an output signal pin that serially transfers data out of the EPCQ device to the FPGA during read or configuration operations. The transition of the signal is on the falling edge of the DCLK signal. During the extended dual input fast write bytes or extended quad input fast write bytes operation, this pin acts as an input signal pin that serially transfers data into the EPCQ device. The data is latched on the rising edge of the DCLK signal. During extended dual input fast read or extended quad input fast read operations, this pin acts as an output signal pin that serially transfer data out of the EPCQ device to the FPGA. The data is shifted out on the falling edge of the DCLK signal. During read, configuration, or program operations, you can enable the EPCQ device by pulling the nCS signal low.
DATA2	9	I/O	For AS x1 mode, extended dual input fast write bytes operation and extended dual input fast read operation, this pin must connect to a 3.3-V power supply. For AS x4 mode, use this pin as an output signal that serially transfers data out of the EPCQ device to the FPGA during read or configuration operations. The transition of the signal is on the falling edge of the DCLK signal. During the extended quad input fast write bytes operation, this pin acts as an input pin that serially transfers data into the EPCQ device. The data is latched on the rising edge of the DCLK signal. During the extended quad input fast read operation, this pin acts as an output signal pin that serially transfers data out of the EPCQ device to the FPGA. The data is shifted out on the falling edge of the DCLK signal.
DATA3	1	I/O	For AS x1 mode, extended dual input fast write bytes operation and extended dual input fast read operation, this pin must connect to a 3.3-V power supply. For AS x4 mode, use this pin as an output signal that serially transfers data out of the EPCQ device to the FPGA during read or configuration operations. The transition of the signal is on the falling edge of the DCLK signal. During the extended quad input fast write bytes operation, this pin acts as an input pin that serially transfers data into the EPCQ device. The data is latched on the rising edge of the DCLK signal. During the extended quad input fast read operation, this pin acts as an output signal pin that serially transfers data out of the EPCQ device to the FPGA. The data is shifted out on the falling edge of the DCLK signal.

Table 5–18. EPCQ Device Pin Description—Preliminary (Part 2 of 2)

Pin Name	Pin Number in 16-Pin SOIC Package	Pin Type	Description
DATA0	15	I/O	For AS x1 mode, use this pin as an input signal pin to write or program the EPCQ device. During write or program operations, the data is latched on the rising edge of the DCLK signal. For AS x4 mode, use this pin as an I/O signal pin. During write or program operations, this pin acts as an input pin that serially transfers data into the EPCQ device. The data is latched on the rising edge of the DCLK signal. During write or program operations, this pin acts as an output signal pin that serially transfers data out of the EPCQ device to the FPGA. The data is shifted out on the falling edge of the DCLK signal. During the extended quad input fast write bytes or extended dual input fast write bytes operations, this pin acts as an input pin that serially transfers data into the EPCQ device. The data is latched on the rising edge of the DCLK signal. During extended dual input fast read or extended quad input fast read operations, this pin acts as an output signal pin that serially transfers data out of the EPCQ device to the FPGA. The data is shifted out on the falling edge of the DCLK signal.
nCS	7	Input	The active low nCS input signal toggles at the beginning and end of a valid operation. When this signal is high, the device is deselected and the DATA pin is tri-stated. When this signal is low, the device is enabled and is in active mode. After power up, the EPCQ device requires a falling edge on the nCS signal before you begin any operation.
DCLK	16	Input	The FPGA provides the DCLK signal. This signal provides the timing for the serial interface. The data presented on the DATA0 pin is latched to the EPCQ device on the rising edge of the DCLK signal. The data on the DATA pin changes after the falling edge of the DCLK signal and is latched in to the FPGA on the next falling edge of the DCLK signal.
V_{CC}	2	Power	Connect the power pins to a 3.3-V power supply.
GND	10	Ground	Ground pin.

Device Package and Ordering Code

This section describes the package offered in EPCQ devices and the ordering codes for each EPCQ device.

Package

The EPCQ128 and EPCQ256 devices are available in 16-pin SOIC packages.

For a 16-pin SOIC package, you can migrate vertically from the EPCQ128 device to the EPCQ256 device. If you use the AS x1 configuration scheme, you can migrate to the EPCS16, EPCS64, or EPCS128 serial configuration devices.

For more information about migration to EPCS, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter.

Ordering Code

Table 5–19 lists the ordering codes for EPCQ devices.

Table 5–19. EPCQ Device Ordering Codes

Device	Ordering Code *(1)*
EPCQ128	EPCQ128SI16N
EPCQ256	EPCQ256SI16N

Note to Table 5–19:

(1) N is lead free.

Document Revision History

Table 5–20 lists the revision history for this chapter.

Table 5–20. Document Revision History

Date	Version	Changes
June 2011	1.0	Initial release.

Section II. Software Settings

Configuration options can be set in the Quartus® II and MAX+PLUS® II development softwares. You can also specify which configuration file formats Quartus II or MAX+PLUS II generates. This section discusses the configuration options available, how to set these options in the software, and how to generate programming files.

This section includes the following chapters:

- Chapter 6, Device Configuration Options
- Chapter 7, Configuration File Formats

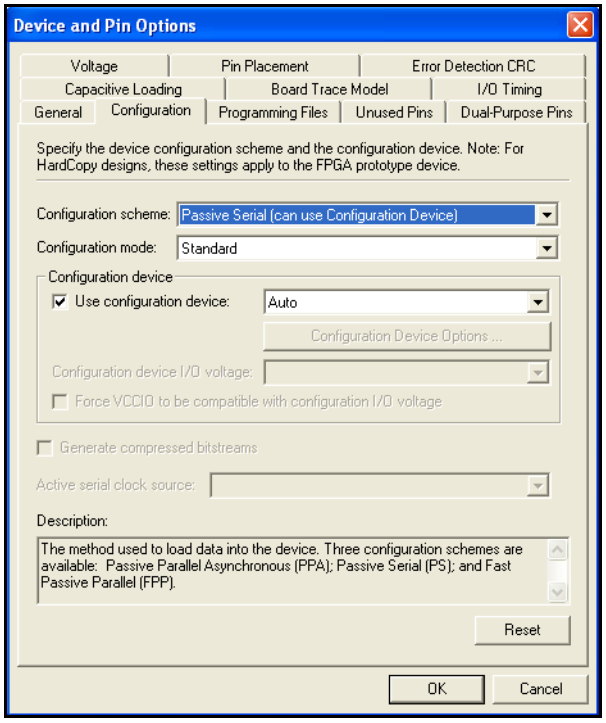
Revision History

Refer to each chapter for its own specific revision history. For information about when each chapter was updated, refer to the Chapter Revision Dates section, which appears in the complete handbook.

CF52006-2.5

Device configuration options can be set in the **Device & Pin Options** dialog box. To open this dialog box, choose **Device** (Assignments menu), then click on the **Device & Pin Options…** button. You can specify your configuration scheme, configuration mode, and your configuration device used (if applicable) in the **Configuration** tab of the **Device & Pin Options** dialog box (Figure 6–1).

Figure 6–1. Configuration Dialog Box



Configuration Scheme

The **Configuration scheme** drop-down list will change with the chosen device family to only show the configuration schemes supported by that device family. The **Configuration mode** selection is only available for devices that support remote and local update, such as Stratix® and Stratix GX devices. If you are not using remote or local update, you should select **Standard** as your configuration mode. For devices that do not support remote or local update, the **Configuration mode** selection will be greyed out.

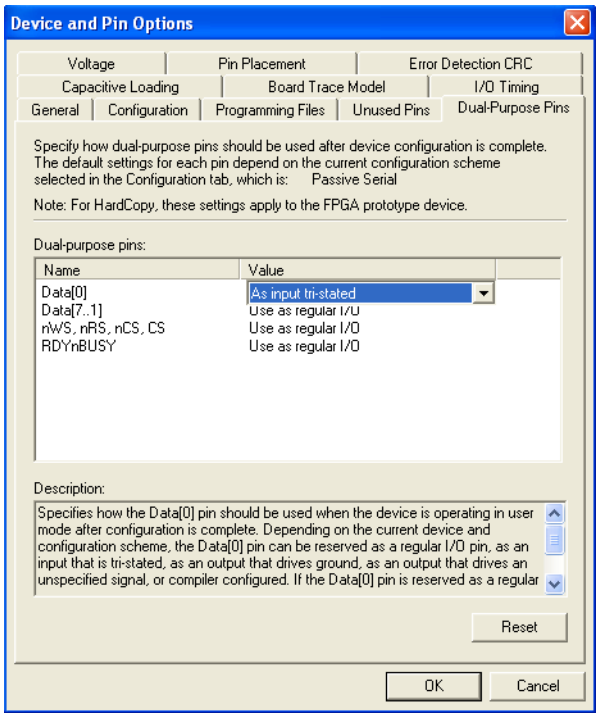
© 2011 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

If you are using a configuration device, turn on **Use configuration device** and specify which configuration device you are using. Choosing the configuration device will direct the Quartus® II compiler to generate the appropriate programming object file (**.pof**). The **Use configuration device** drop-down list will change with the chosen device family to only show the configuration devices that can be used to configure the target device family. If you choose **Auto** as the configuration device, the compiler will automatically choose the smallest density configuration device that fits your design and the **Configuration Device Options...** button will be greyed out.

For more information about configuration device options, refer to the appropriate configuration device data sheet.

You can specify how dual-purpose pins should be used after FPGA configuration is complete through the **Dual-Purpose Pins** tab of the **Device & Pin Options** dialog box. Figure 6–2 shows the **Dual-Purpose Pins** tab for a design that targets a Stratix device and is being configured through PS mode.

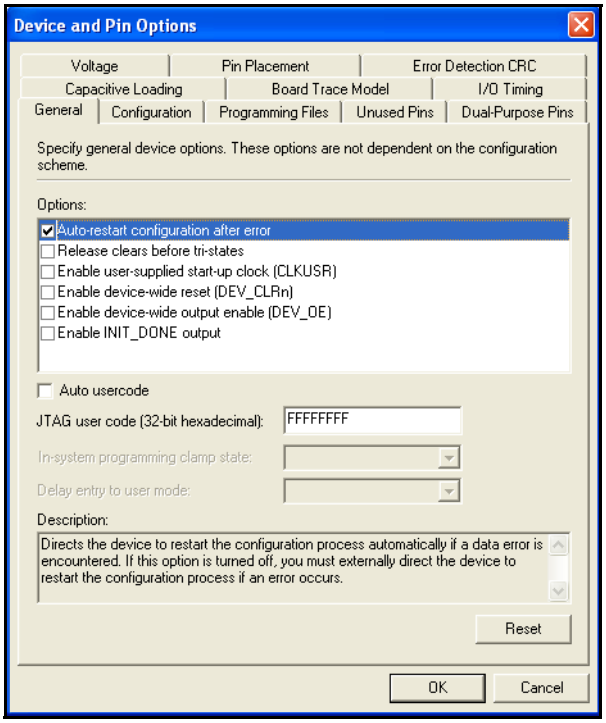
Figure 6–2. Dual-Purpose Pins Dialog Box


Device and Pin Options
Voltage
Pin Placement
Error Detection CRC
Capacitive Loading
Board Trace Model
I/O Timing
General
Configuration
Programming Files
Unused Pins
Dual-Purpose Pins
Specify how dual-purpose pins should be used after device configuration is complete.
The default settings for each pin depend on the current configuration scheme
selected in the Configuration tab, which is: Passive Serial
Note: For HardCopy, these settings apply to the FPGA prototype device.
Dual-purpose pins:
Name
Value
Data[0]
As input tri-stated
Data[7..1]
nWS, nRS, nCS, CS
Use as regular I/O
RDYnBUSY
Use as regular I/O
Description:
Specifies how the Data[0] pin should be used when the device is operating in user mode after configuration is complete. Depending on the current device and configuration scheme, the Data[0] pin can be reserved as a regular I/O pin, as an input that is tri-stated, as an output that drives ground, as an output that drives an unspecified signal, or compiler configured. If the Data[0] pin is reserved as a regular
Reset
OK
Cancel

The drop-down lists will be greyed-out if the pins are not available in the target device family. For the pins that are available, they can be used in one of four states after configuration: as a regular I/O pin, as inputs that are tri-stated, as outputs that drive ground, or as outputs that drive an unspecified signal. If the pin is not used in the mode, the drop-down list will default to the **Use as regular IO** choice. For example, in PS mode, `DATA[7..1]` are not used, therefore the default usage after configuration for these pins is as a regular I/O pin. If the pin is reserved as a regular I/O, the pin can be used as a regular user I/O after configuration.

You can set device options in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box (refer to Figure 6–3).

Figure 6–3. Configuration Options Dialog Box


Device and Pin Options
Voltage
Pin Placement
Error Detection CRC
Capacitive Loading
Board Trace Model
I/O Timing
General
Configuration
Programming Files
Unused Pins
Dual-Purpose Pins
Specify general device options. These options are not dependent on the configuration scheme.
Options:
Auto-restart configuration after error
Release clears before tri-states
Enable user-supplied start-up clock (CLKUSR)
Enable device-wide reset (DEV_CLRn)
Enable device-wide output enable (DEV_OE)
Enable INIT_DONE output
Auto usercode
JTAG user code (32-bit hexadecimal):
FFFFFFFF
In-system programming clamp state:
Delay entry to user mode:
Description:
Directs the device to restart the configuration process automatically if a data error is encountered. If this option is turned off, you must externally direct the device to restart the configuration process if an error occurs.
Reset
OK
Cancel

You can set device options in the MAX+PLUS® II development software by choosing **Global Project Device Options** (Assign menu). Table 6–1 summarizes each of these options.

Table 6–1. Configuration Options (Part 1 of 4)

Device Option	Option Usage	Option On	Option OFF
Auto-restart configuration after error	When a configuration error occurs, the FPGA drives *n*STATUS low, which resets itself internally. The FPGA will release its *n*STATUS pin after a reset time-out period. The *n*STATUS pin is then pulled to V_{CC} by a pull-up resistor, indicating that reconfiguration can begin. You can choose whether reconfiguration is started automatically or manually (by toggling the *n*CONFIG pin).	Reconfiguration starts automatically and the system does not need to pulse *n*CONFIG. After the FPGA releases *n*STATUS and it is pulled to V_{CC} by a pull-up resistor, reconfiguration can begin. In passive configuration schemes that use a configuration device, the FPGA's *n*STATUS pin is tied to the configuration device's OE pin. Hence, the *n*STATUS reset pulse also resets the configuration device automatically. When the FPGA releases *n*STATUS and the configuration device releases its OE pin (that is pulled high), reconfiguration begins. For more information about how long the reset time-out period is, refer to the appropriate device family chapters.	The configuration process stops and to start reconfiguration the system must pulse *n*CONFIG from high-to-low and back high.
Release clears before tri-states	During configuration, the device I/O pins are tri-stated. During initialization, you can choose the order for releasing the tri-states and clearing the registers.	The device releases the tri-states on its I/O pins before releasing the clear signal on its registers.	The device releases the clear signals on its registers before releasing the tri-states. This option allows the design to operate before the device drives out, so all outputs do not start up low.
Enable user-supplied start-up clock (CLKUSR) (APEX™ 20K, APEX II, Cyclone® series, Mercury™, and Stratix series devices only).	This option allows you to select which clock source is used for initialization, either the internal oscillator or external clocks provided on the CLKUSR pin.	The device's internal oscillator (typically 10 MHz) supplies the initialization clock and the FPGA will take care to provide itself with enough clock cycles for proper initialization. The CLKUSR pin is available as a user I/O pin.	The initialization clock must be provided on the CLKUSR pin. This clock can synchronize the initialization of multiple devices. The clock should be supplied when the last data byte is transferred. Supplying a clock on CLKUSR will not affect the configuration process. For more information about how many clock cycles are needed to properly initialize a device, refer to the appropriate device family chapters.

Table 6–1. Configuration Options (Part 2 of 4)

Device Option	Option Usage	Option On	Option OFF
Enable user-supplied start-up clock (CLKUSR) (ACEX® 1K, FLEX® 10K, and FLEX 6000 devices only.)	This option allows you to select which clock source is used for initialization, either external clocks provided on the CLKUSR pin or on the DCLK pin.	In PS and PPS schemes, the internal oscillator is disabled. Thus, external circuitry, such as a configuration device or microprocessor, must provide the initialization clock on the DCLK pin. Programming files generated by the Quartus II or MAX+PLUS II software already have these initialization clock cycles included in the file. In the PPA and PSA configuration schemes, the device's internal oscillator (typically 10 MHz) supplies the initialization clock and the FPGA will take care to provide itself with enough clock cycles for proper initialization. The CLKUSR pin is available as a user I/O pin.	The initialization clock must be provided on the CLKUSR pin. This clock can synchronize the initialization of multiple devices. The clock should be supplied when the last data byte is transferred. Supplying a clock on CLKUSR will not affect the configuration process. For more information about how many clock cycles are needed to properly initialize a device, refer to the appropriate device family chapters.
Enable device-wide reset (DEV_CLRn)	Enables a single pin, DEV_CLRn, to reset all device registers.	Chip-wide reset is not enabled. The DEV_CLRn pin is available as a user I/O pin.	Chip-wide reset is enabled for all registers in the device. All registers are cleared when the DEV_CLRn pin is driven low. The DEV_CLRn pin cannot be used to clear only some of the registers; every device register is affected by this global signal.
Enable device-wide output enable (DEV_OE)	Enables a single pin, DEV_OE, to control all device tri-states.	Chip-wide output enable is not enabled. The DEV_OE pin is available as a user I/O pin.	Chip-wide output enable is enabled for all device tri-states. After configuration, all user I/O pins are tri-stated when DEV_OE is low. The DEV_OE pin cannot be used to tri-state only some of the output pins; every output pin is affected by this global signal.

Table 6–1. Configuration Options (Part 3 of 4)

Device Option	Option Usage	Option On	Option OFF
Enable INIT_DONE output	Enables the INIT_DONE pin, which signals the end of initialization and the start of user-mode with a low-to-high transition.	The INIT_DONE signal is not available. The INIT_DONE pin is available as a user I/O pin.	The INIT_DONE signal is available on the open-drain INIT_DONE pin. When nCONFIG is low and during the beginning of configuration, the INIT_DONE pin will be high due to an external pull-up. When the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low. When initialization is complete, the INIT_DONE pin will be released and pulled high. This low-to-high transition signals the FPGA has entered user mode. The INIT_DONE pin cannot be accessed internally. Therefore, you must loop the INIT_DONE signal back into the device to use the INIT_DONE signal. For more information about the value of the external pull-up resistor, refer to the appropriate device family chapters.
Enable JTAG BST support (FLEX 6000 devices only.)	Enables post-configuration JTAG boundary-scan testing (BST) support in FLEX® 6000 devices.	JTAG BST can be performed before configuration; however, it cannot be performed during or after configuration. During JTAG BST, nCONFIG must be held low.	JTAG BST can be performed before or after device configuration via the four JTAG pins (TDI, TDO, TMS, and TCK); however, it cannot be performed during configuration. When JTAG boundary-scan testing is performed before device configuration, nCONFIG must be held low.

Table 6–1. Configuration Options (Part 4 of 4)

Device Option	Option Usage	Option On	Option OFF
Generate compressed bitstreams (Stratix II and Cyclone series devices only)	Enables Stratix II and Cyclone series FPGAs to receive compressed configuration bitstreams in AS and PS configuration schemes.	The Quartus II software generates uncompressed programming files and Stratix II and Cyclone series FPGAs do not decompress the configuration data.	The Quartus II software generates compressed programming files and Stratix II and Cyclone series FPGAs decompress the bitstream during configuration.
Auto `usercode` (Not available in FLEX 6000 devices.)	Allows you to program a 32-bit user electronic signature into the device during programming (typically for design version control). When the `USERCODE` instruction is loaded into the device, you can shift the signature out of the device.	If this option is off, the JTAG user code option is available and you can specify a 32-bit hexadecimal number for the target device. The JTAG user code is an extension of the option register. This data can be read with the JTAG `USERCODE` instruction.	Uses the checksum value from the SRAM Object File (**.sof**) as the JTAG user code. If this option is on, the JTAG user code option is dimmed to indicate that it is not available.

After enhanced configuration and EPC2 device programming you can choose to automatically configure the targeted FPGAs on board. This can be done by selecting the **Initiate Configuration After Programming** option under the **Programmer** section of the **Options** window (**Tools** menu). This option is similar to issuing the `INIT_CONF` JTAG instruction, which means *n*`INIT_CONF` of the enhanced configuration or EPC2 devices must be connected to the *n*`CONFIG` of the FPGA.

For more information about the `INIT_CONF` JTAG instruction, refer to the *Enhanced Configuration Devices (EPC4, EPC8, and EPC16) Data Sheet* or the *Configuration Devices for SRAM-Based LUT Devices Data Sheet*.

Document Revision History

Table 6–2 lists the revision history for this chapter.

Table 6–2. Document Revision History

Date	Version	Changes
June 2011	2.5	Updated Table 6–1.
December 2009	2.4	■ Updated Figure 6–1, Figure 6–2, and Figure 6–3. ■ Removed “Referenced Documents” section.
October 2008	2.3	■ Added “Referenced Documents” section. ■ Updated new document format.
April 2007	2.2	Added “Chapter Revision History”.
October 2005	2.1	Removed active cross references referring to document outside Chapter 6.
July 2004	2.0	■ Added Stratix II and Cyclone II device information throughout chapter. ■ Updated Default Configuration and Modified Configuration of “auto-restart configuration after error” option in Table 6–1. ■ Added paragraph regarding Initiate Configuration After Programming option on page 6–9.
September 2003	1.0	Initial Release.

7. Configuration File Formats

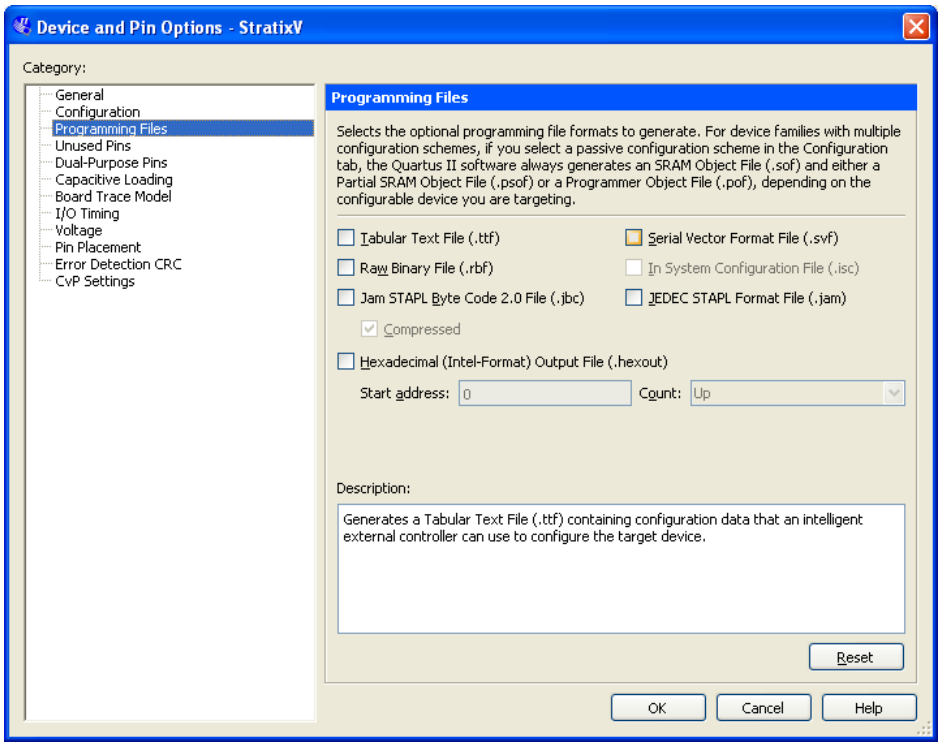
CF52007-2.5

Altera's Quartus® II and MAX+PLUS® II development tools can create one or more configuration and programming files to support the configuration schemes discussed in Volume I. When you compile a design in the Quartus II and MAX+PLUS II software for a device that has programming file support, the software will automatically generate a SRAM Object File (**.sof**) and a Programmer Object File (**.pof**) for a configuration device.

Generating Configuration Files

To instruct Quartus II to generate other configuration file formats during compilation, go to **Programming Files** tab of the **Device & Pin Options** dialog box (refer to Figure 7–1).

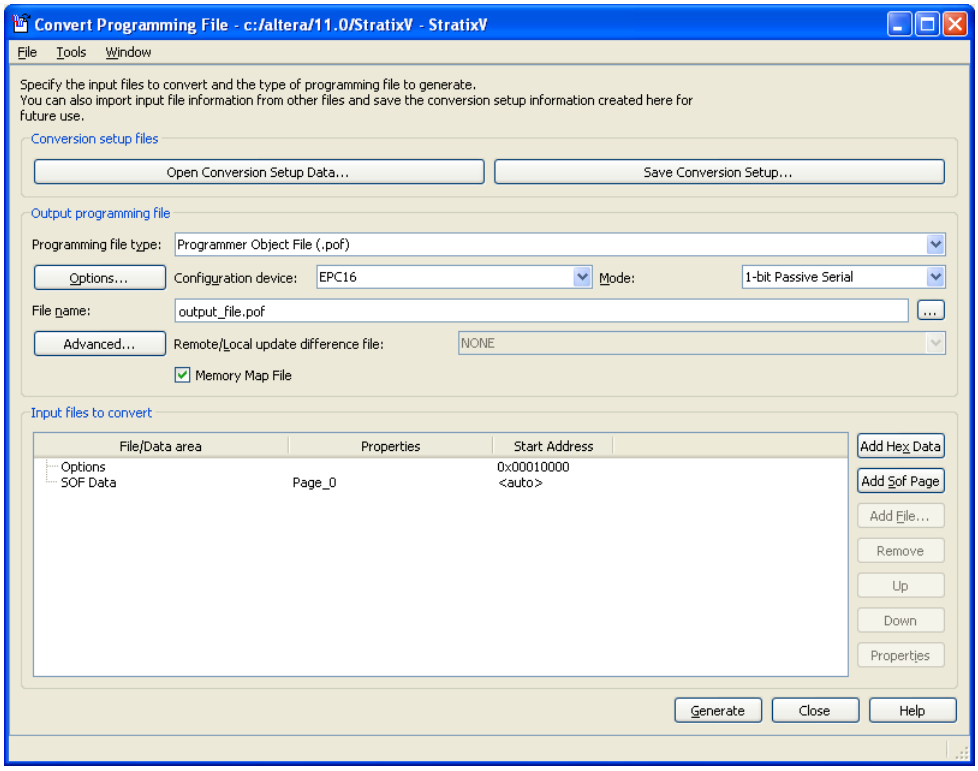
Figure 7–1. Programming Files Dialog Box


Device and Pin Options - StratixV
Category:
General
Configuration
Programming Files
Unused Pins
Dual-Purpose Pins
Capacitive Loading
Board Trace Model
I/O Timing
Voltage
Pin Placement
Error Detection CRC
CvP Settings
Programming Files
Selects the optional programming file formats to generate. For device families with multiple configuration schemes, if you select a passive configuration scheme in the Configuration tab, the Quartus II software always generates an SRAM Object File (.sof) and either a Partial SRAM Object File (.psof) or a Programmer Object File (.pof), depending on the configurable device you are targeting.
Tabular Text File (.ttf)
Serial Vector Format File (.svf)
Raw Binary File (.rbf)
In System Configuration File (.isc)
Jam STAPL Byte Code 2.0 File (.jbc)
JEDEC STAPL Format File (.jam)
Compressed
Hexadecimal (Intel-Format) Output File (.hexout)
Start address:
0
Count:
Up
Description:
Generates a Tabular Text File (.ttf) containing configuration data that an intelligent external controller can use to configure the target device.
Reset
OK
Cancel
Help

© 2011 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

You can also convert **.sof** and **.pof** through the **Convert Programming Files** window (File menu). Figure 7–2 shows an example of the **Convert Programming Files** dialog box set-up to convert **.sof** to a Raw Binary File (**.rbf**).

Figure 7–2. Convert Programming Files Dialog Box



When performing multi-device configuration using a configuration device, you must generate the configuration device's **.pof** from each project's **.sof**. You can combine multiple **.sof** using the Convert Programming Files dialog box in the Quartus II software. The following steps explain how to combine multiple **.sof** into a **.pof**.

1. Choose **Convert Programming Files...** command (File menu).
2. In the **Programming file type** list, choose **Programming Object File** (**.pof**).
3. In the **Configuration device** list, choose the appropriate configuration device.
4. In the **Mode** list, choose the appropriate configuration scheme.
5. You can set configuration devices options by selecting the **Options...** button.
6. Specify the name of the output file in the **File name** box.
7. In the **Input files to convert** box, click on **SOF Data**, so that the **Add File…** button becomes active.
8. Click on the **Add File…** button and select the **.sof** to be converted. This step can be repeated to combine multiple **.sof** into a single **.pof**. The order of the **.sof** should match the order of the devices in the chain.
9. Click **OK**.

10. When generating multiple **.pof** for EPC1 or EPC2 devices, the first device's **.pof** name will be as specified, while the second device's **.pof** name will have a "_1" extension (e.g., **top_1.pof**)

When performing multi-device configuration using an external host, such as a microprocessor or CPLD, you should generate one combined configuration file from each project's **.sof**. You can combine multiple **.sof** using the **Convert Programming Files** dialog box in the Quartus II software. The following steps explain how to combine multiple **.sof** into one configuration file.

1. Choose the **Convert Programming Files...** command (File menu).
2. In the **Programming file** type list, choose the appropriate file format (Hexadecimal (Intel-Format) Output File for SRAM (**.hexout**), **.rbf**, or Tabular Text File (**.ttf**)).
3. In the **Mode** list, choose the appropriate configuration scheme.
4. Specify the name of the output file in the **File name** box.
5. In the **Input files to convert** box, click on **SOF Data**, so that the **Add File…** button becomes active.
6. Click on the **Add File…** button and select the **.sof** to be converted. This step can be repeated to combine multiple **.sof** into one configuration file. The order of the **.sof** (from top to bottom) should match the order of the devices in the chain.
7. Click **OK**.

The following steps explain how to convert a **.sof** for ACEX® 1K, FLEX® 10K, or FLEX 6000 devices using the MAX+PLUS II software.

1. In the MAX+PLUS II Compiler or Programmer, choose the **Convert SRAM Object Files** command (File menu.)
2. In the **Convert SRAM Object Files** dialog box, click on the **Select Programming File…** button to specify which **.sof** to convert. This step can be repeated to combine multiple **.sof** into one configuration file. The order of the **.sof** (from top to bottom) should match the order of the devices in the chain.
3. Specify the name of the output file in the **File Name** box.
4. Choose the appropriate configuration file format through the **File Format** pull-down list.
5. Click **OK**.

The following sections give a description of the supported configuration file formats.

SRAM Object File (.sof)

You should use a **.sof** during PS configuration when the configuration data is downloaded directly to the FPGA using the Quartus II or MAX+PLUS II software with a USB Blaster, MasterBlaster™, ByteBlaster™ II, EthernetBlaster or ByteBlasterMV™ cable. The Quartus II and MAX+PLUS II compiler automatically generates the **.sof** for your design. When using a **.sof**, the Quartus II or MAX+PLUS II software controls the configuration sequence and automatically inserts the appropriate headers into the configuration data stream. All other configuration files are created from the **.sof**.

Programmer Object File (.pof)

A **.pof** is used by the Altera® programming hardware to program a configuration device. The Quartus II and MAX+PLUS II compiler automatically generate a **.pof** for your design. For smaller devices (e.g., EPF10K20 devices), multiple **.sof** can fit into one configuration device; for larger devices (e.g., APEX 20K devices), multiple configuration devices may be required to hold the configuration data.

Raw Binary File (.rbf)

The **.rbf** is a binary file containing the configuration data. The **.rbf** does not contain byte separators (e.g. commas or carriage returns); it is literally a raw binary file that contains a binary bitstream of configuration data. For example, one byte of **.rbf** data is 8 configured bits `10000101` (85 Hex). Data must be stored so that the least significant bit (LSB) of each data byte is loaded first. The converted image can be stored on a mass storage device. The microprocessor can then read data from the binary file and load it into the FPGA. You can also use the microprocessor to perform real-time conversion during configuration. In the PS configuration schemes, each byte of data should be sent with LSB first. In the FPP, PPS, and PPA configuration schemes, the target device receives its information in parallel from the data bus, a data port on the microprocessor, or some other byte-wide channel.

For more information about creating **.rbf**, search for "RBF" in Quartus II or MAX+PLUS II Help.

Raw Programming Data File (.rpd)

The **.rpd** file is a binary file for EPCS devices containing a binary bitstream of configuration data for FPGAs that support Active Serial configuration. This file is stored in the serial configuration devices in an embedded environment outside the Quartus II software. The FPGA can then be configured with the Active Serial (AS) configuration scheme where the FPGA loads the **.rpd** file stored in the serial configuration device. The **.rpd** file size is equal to the memory size of the targeted serial configuration device. A **.rpd** file can only be generated from a **.pof** in the **Convert Programming Files** dialog box (File menu).

The **.rpd** file is different from the **.rbf**, even for a single device configuration file. In multi-device chains, the **.rpd** file is not the concatenation of the corresponding **.rbf**. The LSB of each byte in the **.rpd** file should be written to the serial configuration device first.

For more information about creating **.rpd** files, search for "RPD" in Quartus II Help or refer to the *AN 418: SRunner: An embedded Solution for Serial Configuration Device Programming*.

Hexadecimal (Intel-Format) File (.hex) or (.hexout)

A **.hex** file is an ASCII file in the Intel HEX format. Microprocessors or external hosts can use the **.hex** file to store and transmit configuration data using the configuration schemes supported by microprocessors. This file can also be used by third-party programmers to program Altera's configuration devices.

For more information about creating **.hex** files, search for "Hex File" in Quartus II or MAX+PLUS II Help.

Tabular Text File (.ttf)

The **.ttf** is a tabular ASCII file that provides a comma-separated version of the configuration data for the FPP, PPS, PPA, and bit-wide PS configuration schemes. In some applications, the storage device containing the configuration data is neither dedicated to nor connected directly to the target device. For example, a configuration device can also contain executable code for a system (e.g., BIOS routines) and other data. The **.ttf** allows you to include the configuration data as part of the microprocessor's source code using the include or source commands. The microprocessor can access this data from a configuration device or mass-storage device and load it into the target device. A **.ttf** can be imported into nearly any assembly language or high-level language compiler.

For more information about creating **.ttf**, search for "TTF" in Quartus II or MAX+PLUS II Help.

Serial Bitstream File (.sbf)

A **.sbf** is used in PS schemes to configure FLEX 10K and FLEX 6000 devices in-system with the BitBlaster™ cable.

The BitBlaster is obsolete. **.sbf** is supported by the MAX+PLUS II software only.

For more information about creating **.sbf**, search for "SBF" in MAX+PLUS II Help.

Jam File (.jam)

A **.jam** file is an ASCII text file in the Jam device programming language that stores device programming information. These files are used to program, verify, and blank-check one or more devices in the Quartus II or MAX+PLUS II Programmer or in an embedded processor environment.

For more information about creating **.jam** files, search for "Jam" in Quartus II or MAX+PLUS II Help.

Jam Byte-Code File (.jbc)

A **.jbc** file is a binary file of a **.jam** file in a byte-code representation. **.jbc** files store device programming information used to program, verify, and blank-check one or more devices in the Quartus II or MAX+PLUS II Programmer or in an embedded processor environment.

For more information about creating **.jbc** files, search for "JBC" in Quartus II or MAX+PLUS II Help.

Document Revision History

Table 7–1 lists the revision history for this chapter.

Table 7–1. Document Revision History

Date	Version	Changes
June 2011	2.5	Updated Figure 7–1 and Figure 7–2.
December 2009	2.4	■ Updated Figure 7–1 and Figure 7–2. ■ Removed "Referenced Documents" section.
October 2008	2.3	■ Updated "Raw Programming Data File (.rpd)" section. ■ Added "Referenced Documents" section. ■ Updated new document format.
April 2007	2.2	Added document revision history.
August 2005	2.1	Removed active cross references referring to document outside Chapter 7.
July 2004	2.0	■ Added paragraph regarding difference of .**rpd** from .**rbf** in the "Raw Programming Data File (.rpd)" section.
September 2003	1.0	Initial Release.

Section III. Advanced Configuration Schemes

This section discusses configuring configuration chains that contain a mixture of Altera® device families, combining different configuration schemes on your board and using a CPLD and flash memory to configure your Altera FPGA. It is recommended that you read the chapters in Volume I for your target device family before reading this section.

This section includes the following chapter:

- Chapter 8, Combining Different Configuration Schemes

Revision History

Refer to each chapter for its own specific revision history. For information about when each chapter was updated, refer to the Chapter Revision Dates section, which appears in the complete handbook.

8. Combining Different Configuration Schemes

CF52009-2.5

This chapter describes how to configure Altera® FPGAs using multiple configuration schemes on the same board. Combining JTAG configuration with passive serial (PS) or active serial (AS) configuration on your board is useful in the prototyping environment because it allows multiple methods to configure your FPGA. For example, if your production environment calls for PS configuration using a configuration device, you must reprogram your configuration device every time you wanted to test a design change in your FPGA. If you include the FPGA in the same JTAG chain as the configuration device, the FPGA can be reconfigured via JTAG without having to reprogram the configuration device.

In this chapter, the generic term "download cable" includes the Altera USB Blaster universal serial bus (USB) port download cable, MasterBlaster™ serial/USB communications cable, EthernetBlaster, ByteBlaster™ II parallel port download cable, and the ByteBlasterMV™ parallel port download cable. In this section, the generic term "FPGA" includes ACEX® 1K, APEX™ 20K, APEX II, Arria® series, Cyclone® series, FLEX® 10K, Mercury™, and Stratix® series devices.

The figures in this chapter will only show the configuration interface connections.

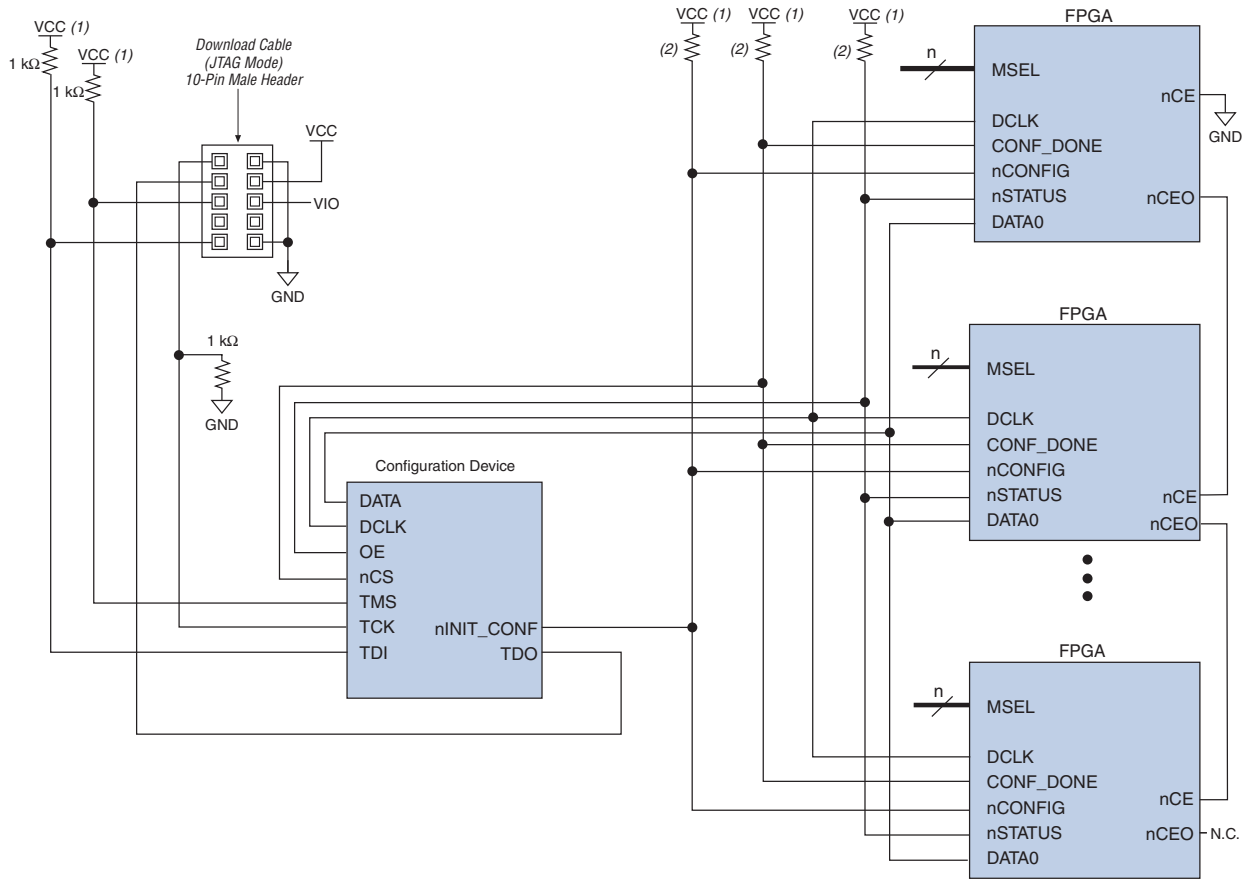
For detailed information about pull-up resistor values or other pins on the specific FPGA or configuration device, refer to appropriate chapter in the *Configuration Handbook*.

© 2009 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX are Reg. U.S. Pat. & Tm. Off. and/or trademarks of Altera Corporation in the U.S. and other countries. All other trademarks and service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

Passive Serial and JTAG

Figure 8–1 shows the configuration interface connections when you are using a download cable to JTAG program a configuration device and the configuration device is used to configure the FPGAs. In Figure 8–1, multiple FPGAs are daisy-chained together and the MSEL pins should be set to select PS as the configuration mode.

Figure 8–1. JTAG Programming of Configuration Device with PS Configuration of FPGA Using a Configuration Device



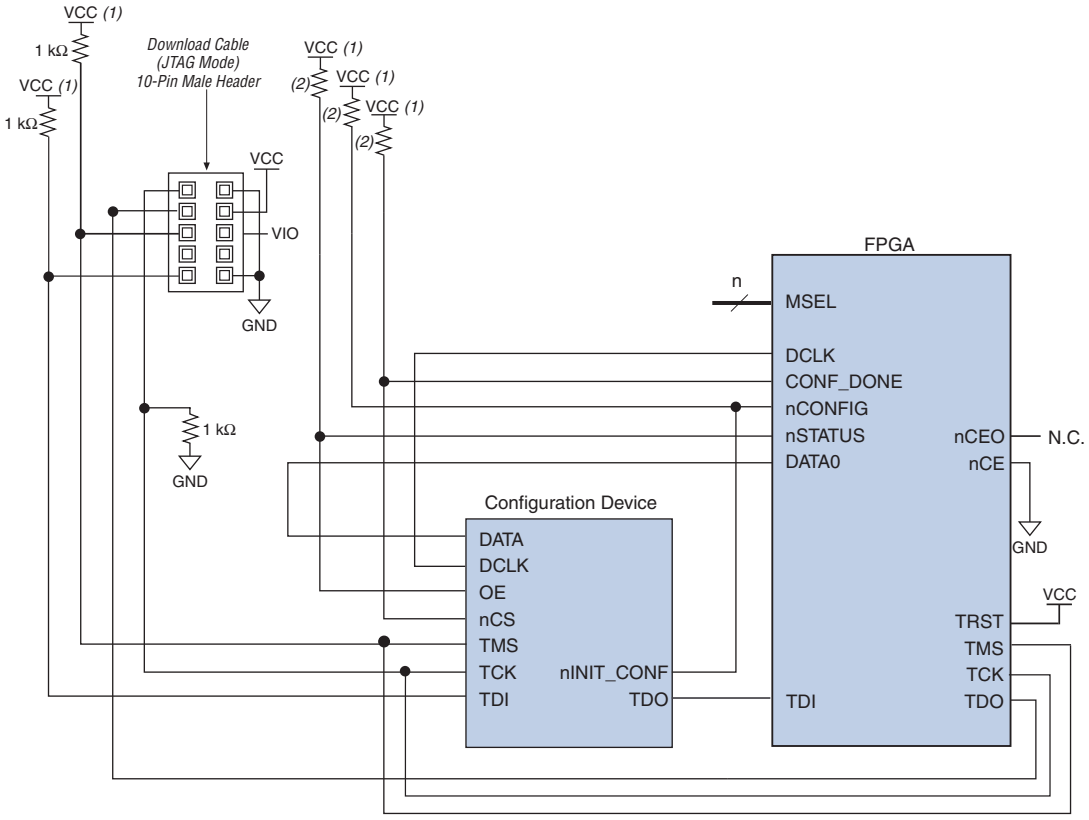
Notes to Figure 8–1:

(1) V_{CC} should be connected to the same supply voltage as the configuration device. For APEX 20KE devices, nCONFIG should be pulled up to V_{CCINT}.
(2) If the internal pull-up resistors of the configuration device are used, external pull-up resistors should not be used on these pins.

Figure 8–2 shows the configuration interface connections when the configuration device and the FPGA are in the same JTAG chain. Make sure the TDO signal drives out a high enough voltage to meet the next device's TDI minimum high-level input voltage (V_{IH}). The TDO output will drive out the voltage of the I/O bank's V_{CCIO} where it resides. For example, if the TDO pin resides in an I/O bank whose V_{CCIO} is set to 3.3 V, the TDO pin will drive out 3.3 V. The download cable is used to JTAG program the configuration device and the FPGA. The configuration device is used to configure the FPGA. The MSEL pins should be set to select PS as the configuration mode.

If there is a configuration device on board, upon power-up you should allow the FPGA to finish configuration before attempting JTAG configuration.

Figure 8–2. JTAG Programming of Configuration Device and FPGA with PS Configuration of FPGA Using a Configuration Device



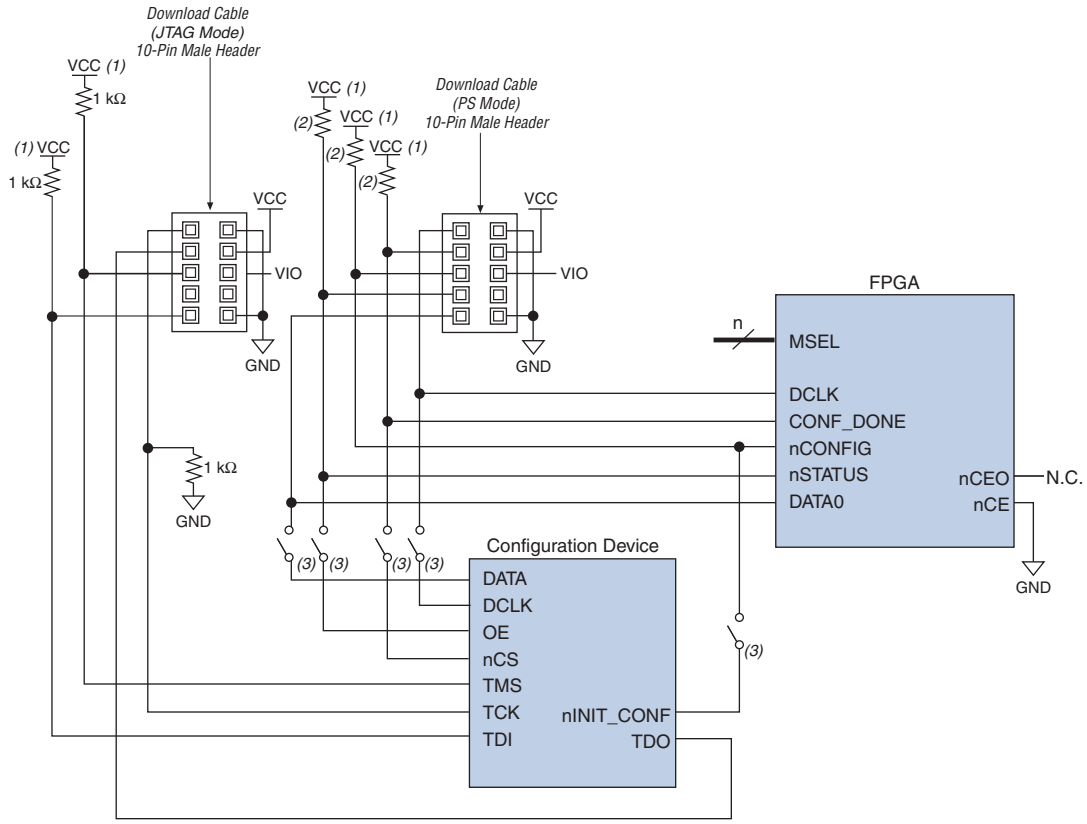
Notes to Figure 8–2:

(1) V_{CC} should be connected to the same supply voltage as the configuration device. For APEX 20KE devices, `nCONFIG` should be pulled up to V_{CCINT}.

(2) If the internal pull-up resistors of the configuration device are used, external pull-up resistors should not be used on these pins.

The download cables can be used in different modes (e.g., JTAG mode or PS mode) and in each mode, the header of the download cable connects to different pins on the FPGA. Therefore, two separate 10-pin headers are required on your board in order to support two different modes of the download cable. Figure 8–3 shows a schematic with two download cables. One download cable is used in JTAG mode to JTAG program the configuration device. The second download cable is used in PS mode to configure the FPGA using PS configuration. The MSEL pins should be set to select PS as the configuration mode.

Figure 8–3. JTAG Programming of Configuration Device with PS Configuration of FPGA Using a Configuration Device and Download Cable



Notes to Figure 8–3:

(1) V_{CC} should be connected to the same supply voltage as the configuration device. For APEX 20KE devices, nCONFIG should be pulled up to V_{CCINT}.

(2) If the internal pull-up resistors of the configuration device are used, external pull-up resistors should not be used on these pins.

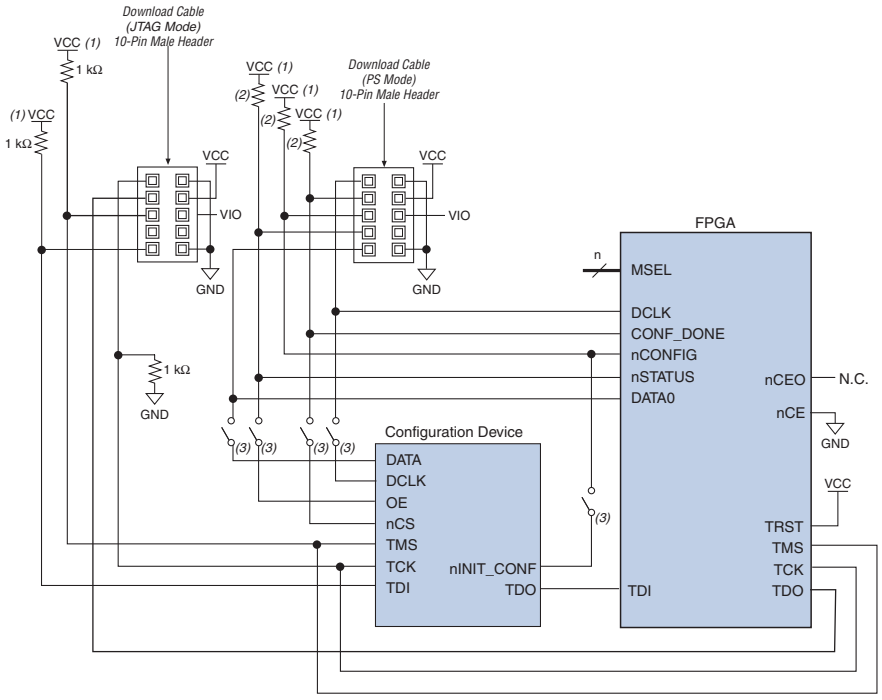
(3) To configure the FPGA with a download cable, you should either remove the configuration device from its socket or place a switch on the five common signals between the download cable and the configuration device.

You should not attempt PS configuration with a download cable while a configuration device is connected to an FPGA.

If you try to configure the FPGA using the download cable while the configuration device is connected to the FPGA, the low signals driven on the *n*STATUS and CONF_DONE pins will pull the OE and *n*CS pins of the configuration device low. This will reset the configuration device and cause it to try to configure the FPGA. To perform PS configuration with the download cable, you should either remove the configuration device from its socket when using the download cable, or place a switch on the five common signals between the download cable and the configuration device.

Figure 8–4 shows a schematic that allows configuration of the FPGA with either a PS mode download cable or JTAG mode download cable. Additionally, the FPGA can be configured using the configuration device. One download cable is used in JTAG mode to JTAG program the configuration device and FPGA. In Figure 8–4 the configuration device and FPGA are in the same JTAG chain. Make sure the TDO signal drives out a high enough voltage to meet the next device's TDI minimum high-level input voltage (V_{IH}). The TDO output will drive out the voltage of the I/O bank's V_{CCIO} where it resides. For example, if the TDO pin resides in an I/O bank whose V_{CCIO} is set to 3.3 V, the TDO pin will drive out 3.3 V. The second download cable is used in PS mode to configure the FPGA using PS configuration. The MSEL pins should be set to select PS as the configuration mode.

Figure 8–4. Combining JTAG Programming of Configuration Device and FPGA with PS Configuration of FPGA Using a Configuration Device and Download Cable



Notes to Figure 8–4:

(1) V_{CC} should be connected to the same supply voltage as the configuration device. For APEX 20KE devices, *n*CONFIG should be pulled up to V_{CCINT}.

(2) If the internal pull-up resistors of the configuration device are used, external pull-up resistors should not be used on these pins.

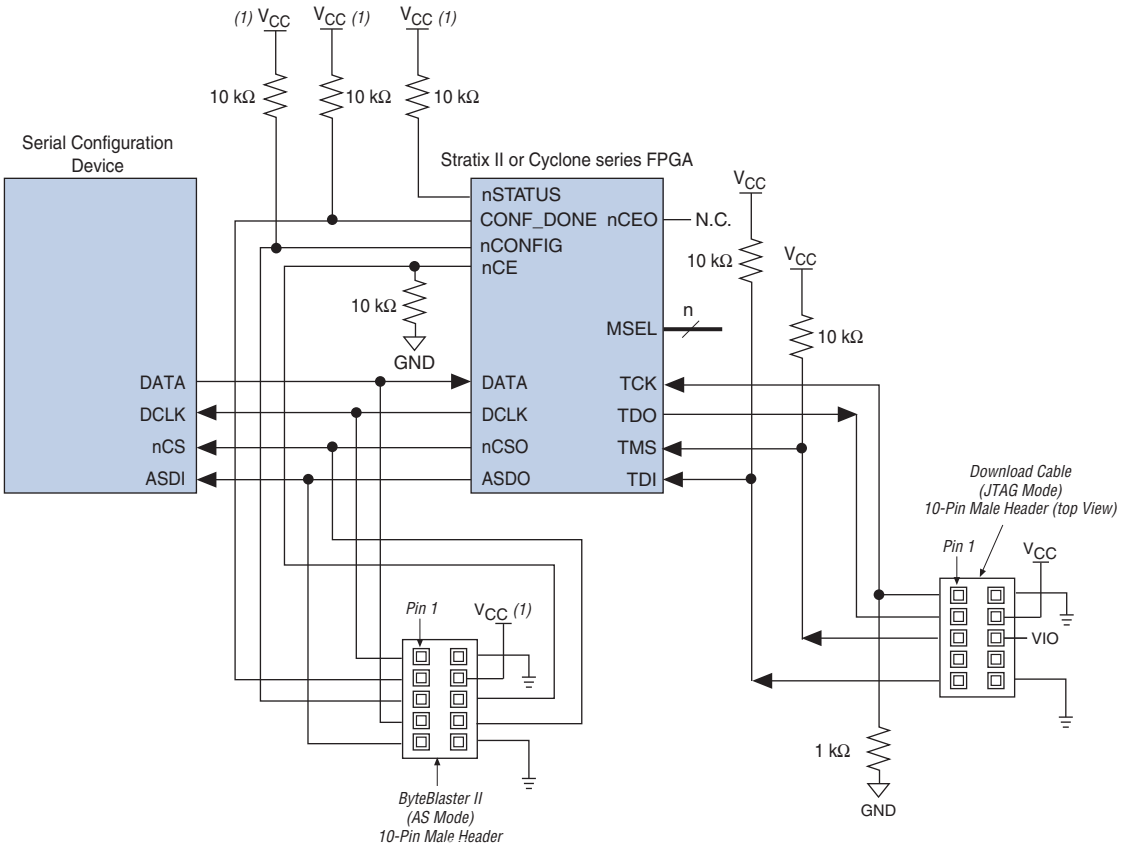
(3) To configure the FPGA with a download cable, you should either remove the configuration device from its socket or place a switch on the five common signals between the download cable and the configuration device.

Figure 8–1 on page 8–2 and Figure 8–4 also apply for fast passive parallel (FPP) mode, except DATA[7..0] is connected from the enhanced configuration device to the FPGAs that support FPP configuration. The MSEL pins must be set accordingly.

Active Serial and JTAG

For devices that support AS configuration (e.g. Stratix series, Cyclone series, or Arria series devices), you can combine the AS configuration scheme with JTAG-based configuration (refer to Figure 8–5). This setup uses two 10-pin download cable headers on the board. One download cable is used in JTAG mode to configure the Stratix II or Cyclone series FPGA directly via the JTAG interface. The other download cable is used in AS mode to program the serial configuration device in-system via the AS programming interface. The MSEL pins should be set to select the AS configuration mode. If you try configuring the device using both schemes simultaneously, JTAG configuration takes precedence and AS configuration will be terminated.

Figure 8–5. Combining JTAG Programming of Configuration Device and FPGA with AS Configuration of FPGA Using a Configuration Device and Download Cable



Note to Figure 8–5:

(1) V_{CC} should be connected to 3.3 V.

Document Revision History

Table 8–1 lists the revision history for this chapter.

Table 8–1. Chapter Revision History

Date	Version	Changes
December 2009	2.5	■ Updated Table 8–1. ■ Removed "Referenced Documents" section.
October 2008	2.4	■ Updated "Introduction" and "Active Serial and JTAG" sections. ■ Added "Referenced Documents" section. ■ Updated new document format.
January 2008	2.3	Corrected figure title in Figure 9–5.
April 2007	2.2	Added document revision history.
August 2005	2.1	Removed active cross references referring to document outside Chapter 9.
July 2004	2.0	Added Stratix II and Cyclone II device information throughout chapter.
September 2003	1.0	Initial Release.